國立臺東大學

執行教育部 109 年度高等教育深耕計畫

2020 打造綠色國際大學

校園示範點校園創作成果報告書

議題：C2-1 ：

VR 棒球遊戲結合腦波觀測之應用

主持人：資管系 謝明哲 副教授

學生： 資管四 張元豪

資管四 李維

資管四 洪久洲

資管四 温舜傑

中華民國 110 年 6 月 8 日

摘要

近幾年虛擬實境的應用在眾多領域中蓬勃發展起來，許多行業開始利用虛擬實境設備進行員工教育訓練，不僅可以降低訓練的成本，也可以避免一些訓練意外發生。本論文主要在探討利用虛擬實境進行棒球訓練，並結合腦波儀的觀測，取得並分析訓練者的身心狀況。

關鍵詞：虛擬實境、棒球訓練、腦波、身心狀況

誌謝

這次的專題能呈現出來，首先要感謝給予我們指導的師長以及學長姐。感謝館長，在專題的走向以及設計上給了我們許多建議。感謝李偉勝學長給了我們技術上的建議以及支援。

專題第九小組 謹誌於

國立臺東大學

資訊管理學系

中華民國一〇九年六月

目錄

圖目錄

表目錄

第一章 前言

1.1 背景與動機

棒球是一種團體球類運動，由人數最少為 9 人的兩支伍在一個扇形的球場進行攻擊與守備。棒球球員分為攻、守兩方攻方球員利用球棒將守方投擲的球擊出，隨後沿著四個壘位進行跑壘，當成功跑一圈回到本壘就可得 1 分；而守方則利用手套將攻方擊出的球接住或擲回將攻方球員打出局。比賽中，兩隊輪流攻守，九局中（少棒為六局，青少棒為七局）得分較高的一隊一定會勝出。若正規 9 局打完後雙方得分仍相同，則進入延長賽，而棒球也是非常受歡迎的球類運動，世界各地都有許多人熱愛棒球。多年來，台灣已經培育出為數眾多的棒球選手，下至少棒，上至美國職棒，都可以見到國人在球場上的英姿。在國際賽事上，棒球不僅僅是一項運動競賽，它更可以凝聚國人向心力。每當中華隊在國際賽中取得佳績時，在社群網站或是街道上總是能聽到他人談論中華隊在球賽中的英姿。

而近年來大家對於娛樂遊戲越趨重視，加上網路興起，Virtual Reality、Augmented Reality 的問世，造成電子遊戲變得十分流行甚至有遊戲是生活中不可或缺的趨勢，Virtual Reality 與 Augmented Reality 在未來擁有很大的發展潛力，加上系上有學長姊曾以 Virtual Reality 當主題，也因為我們對 Virtual Reality 技術也有興趣，所以選擇以 Virtual Reality 遊戲結合棒球打擊來做為此次專題主題。

1.2 目的

棒球屬於戶外運動，在下雨頻繁的台灣，難免會受到天氣的影響導致比賽為此增加危險性甚至於中斷，而在偏鄉地區，沒有足夠場地與棒球相關設施是家常便飯的困難，且交通不便也是主要問題之一。因此，我們將設計一套棒球遊戲讓使用者能選擇球種球速，並結合腦波來訓練球員打擊時的內心情況，提高專注度、減低緊張造成的

失誤，既不需要實際球場也不必在意交通所帶來的問題，從打擊方面開始著手設計，我們將使用虛擬實境(VR)來達成擬真的目的。

因此透過這次研究希望能達成以下目的：

(1) 解決人數不足的問題

(2) 克服天候因素

(3) 客製化訓練

(4) 訓練球員內心情況，提高專注度、減低緊張造成的失誤

1.3 步驟與方法

本組在此研究專題前並無 VR、AR 之相關技術，訂定此題目是以興趣及學長姐之題目作為考量，所以並非在一開始就有完整的構想，而是採用敏捷式開發，在討論過程中逐步新增及修改內容，經過小組討論後我們覺得能讓使用者進行流暢的打擊，並能自由選擇球種、球速並結合腦波以利於訓練是非常好的，在此部份我們有參考例如：Unity 3D 教學影片、C#教學資料圖書、人物創建場景創建等等，並在此研究當中不斷嘗試與更新程式方法，讓使用者在遊戲過程中能達成訓練目的。

本組讓使用者戴上 HTC VIVE 後，能看到操作面板(能自由開啟或關閉)並能以此操作面板選擇球種與球速，打擊完成後能顯示腦波數據以此來調整專注度，情緒的起伏影響，藉此提高打擊精準度。

1.4 範圍與限制

因為本研究的設備、操作環境、經費等考量下，仍有部分限制存在，詳細說明情況如下：

(1) 整個實驗室過程受限於虛擬實境設備的可移動最大範圍與本實驗室，不能超過。

(2) 現在 VR 設備商家不只一家，本實驗 VR 設備設限為 HTC 的 VR 設備：HTC VIVE。

(3) 因為棒球打擊是會有真實手感的，但由於設備及技術的限制，導致模擬真實手感並不容易可說是十分困難，所以打擊時會與現實擊球的情景有落差。

第二章 文獻探討

2.1 相關實境介紹

2.1.1 虛擬實境(VR)

虛擬實境(Virtual Reality, VR)是利用電子設備模擬產生一個三維空間的虛擬空間，透過虛擬空間提供給使用者在視覺跟聽覺的感官模擬，並且與虛擬的世界互動，達到身歷其境，感同身受。

而依照 Krueger(1991)提出虛擬實境具有「3i」特性:

(1) 沉浸感(Immersion)：讓使用者透過如：虛擬實境眼鏡、頭盔和基地台等設備，完全融入虛擬世界中，使五感(視覺、聽覺、觸覺、嗅覺和味覺)能充分表現，達到與現實世界一樣的臨場感。

(2) 互動性(interaction)：虛擬裝置透過 3D 立體影像顯示技術，讓使用者能夠在虛擬世界中能夠充分與畫面互動，體驗如真實世界一樣的互動。

(3) 想像性(Imagination)：使用者進入虛擬場景中的任何感官效果，如光線、聲音、震動等，能刺激使用者的想像力，並可以營造另一個想像空間，達到虛擬的效果。

2.1.2 擴增實境(AR)

擴增實境(Augmented Reality，AR) 是由 VR 所衍生出來的一種技術，它是一種將虛擬資訊擴增到現實空間中的技術，它所強調的不是要取代現實空間，而是在現實空間中添加一個虛擬物件，藉由攝影機影像、圖像分析技術與電腦程式的結合，當設定好的圖片出現在鏡頭裡面，就會出現對應的虛擬物件。

以 2016 年推出的 Pokémon Go「寶可夢」遊戲為例

透過智慧型手機的相機拍攝真實世界的影像，接著經由 App 將神奇寶貝影像放置在手機螢幕上，製造出真實與虛擬之間的融合，讓玩家體驗互動性。

2.1.3 混合實境(MR)

混合現實（Mixed Reality，MR）也稱作擴增虛擬，也就是把現實世界與虛擬世界合併再一起並建立出一個新的環境以及符合一般視覺上所認知的虛擬影像，讓虛擬世界中也有現實世界的物件並且即時的產生互動。

圖 1 混合實境MR

2.2 相關技術與軟硬體設備

2.2.1 虛擬實境硬體設備

1.HTC VIVE 頭戴式顯示器(HMD)

2.一對無線手把控制器

3.一對紅外線基地台(Lighthouse or Base Station)

4.配有獨立顯示卡之個人電腦

虛擬實境最重要的就是設備而當中做重要的莫過於紅外線基地台 Lighthouse，

Lighthouse 會發出紅外線照射整個遊玩空間，使控制器和 HMD 可以得到一個座標，知道控制器和 HMD 在遊玩空間中的位置，相當於創建一個空間並定位，使得使用者在現實空間中移動時在虛擬世界裡也會同時移動。

Lighthouse 系統的運作方式是：

1. Lighthouse 中固定的紅外線 LED 發出一陣閃光。

2. 控制器和 HMD 上的紅外線感應器偵測到閃光開始計時。

3. Lighthouse 中垂直轉動的紅外線雷射掃射整個遊玩空間。

4. 控制器和 HMD 上的紅外線感應器偵測到雷射結束計時，依據時間的長短便可以得知垂直角度。

步驟五至步驟八和步驟一到四大致相同，不同之處在於將「垂直轉動的紅外線雷射」換成「水平轉動的紅外線雷射」，得知水平角度。

9. 取得控制器或 HMD 上三個以上紅外線感應器相對於 Lighthouse 的垂直、水平角度後，由於每個紅外線感應器之間的相對位置是固定的，因此能夠推算出控制器或 HMD 在空間中的位置。

10. 步驟 1.~9.每一秒會重複執行數十次，因此即使快速揮舞 Vive 控制器，Lighthouse 系統依舊能夠準確地追蹤控制器所在的位置。

圖 2 紅外線基地台Lighthouse

2.2.2 開發工具

目前 VR 以及 3D 的開發環境中，有許多的開發引擎可以選擇，主流的有 Unity 以及 UnrealEngine 可供選擇。本次的研究我們將以 Unity 來做為開發引擎。理由是 Unity 在使用上較容易上手，在網路上有許多教學以及第三方資源，且開發的成本也相對較低。

2.2.3 程式語言

在 Unity 3D 中，我們使用的程式語言為 C#，在 Unity 3D 中可以直接在 Project→Assets 裡的空白處又見新增一項 C#腳本，而底下為一新增空白腳本，Start()中的程式代表遊戲一開始造成的影響，而 Update()中則是影響遊戲進行直到遊戲結束為止。

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   public class NewBehaviourScript : MonoBehaviour
6   {
7       // Start is called before the first frame update
8       void Start()
9       {
10
11      }
12
13      // Update is called once per frame
14      void Update()
15      {
16
17      }
18  }
19
```

圖 3 空白腳本範例

而在 Unity 3D 中也有屬於 Unity 自己的文法用法，在網路上有提供用法說明，可在下圖中查找所需要的程式。

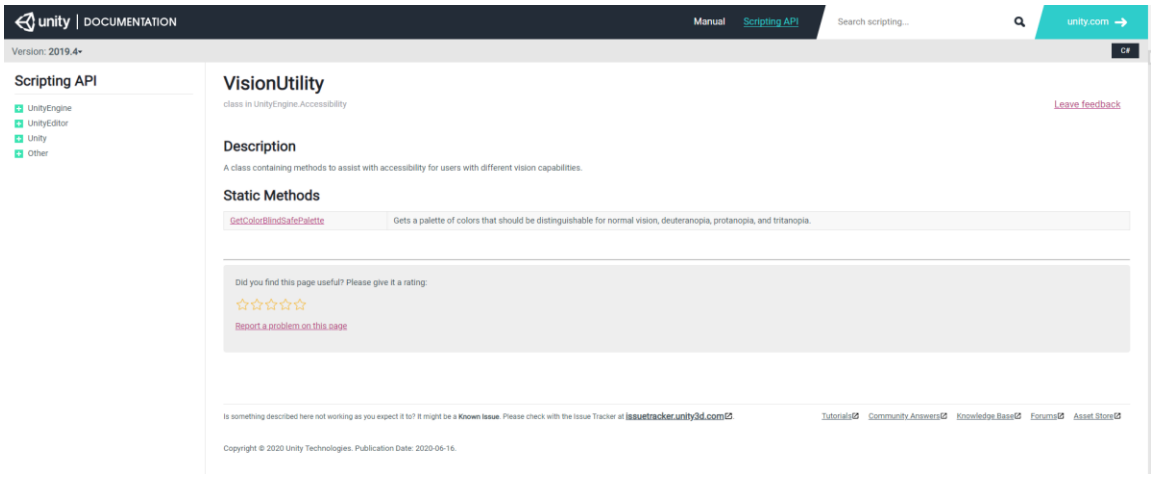


圖 4 Umity API

2.3 相關領域知識

2.3.1 腦波

人類的大腦裡，有許多神經細胞日復一日不斷活動；細胞活動會發出電磁波在科學儀器上，看起來就像波動一樣，故稱之為腦波。腦波依頻率可分為五大類：β波（顯意識 14-30HZ）、α波（橋樑意識 8-14HZ）、θ波（潛意識 4-8Hz）及δ波（無意識 4Hz 以下）和γ波(專注於某件事 30HZ 以上)，本研究主要探討 Alpha（α）Beta（β）Gamma（γ）三種波。

腦波種類		頻率	特性
Alpha（α）	慢速α波	8-9 HZ	臨睡前意識逐漸模糊
	中間α波	9-12 HZ	靈感、直覺或點子爆發狀態 身心輕鬆而注意力集中
	快速α波	12-14 HZ	高度警覺狀態
Beta（β）	Low Range	12.5-16 HZ	放鬆但精神集中
	Middle Range	16.5-20 HZ	思考、處理接收到外界資訊
	High Range	20.5-28 HZ	激動、焦慮
Gamma（γ）		25-100 HZ	提高意識、幸福感、減輕壓力

表格 1 腦波種類

2.3.2 腦波設備

BrainLink Lite 是深圳市宏智力科技有限公司開發的一款智能穿戴硬體，它是一個頭戴式腦電波傳感設備。BrainLink 通過藍牙無線連結手機、平板電腦等終端設備。配合相應的應用軟體就可以實現大腦狀態可視化，腦能力訓練甚至意念力互動操控。

人類對腦電波的探索已經超過了半個世紀。簡單說，人體是一部相當精密的儀器，人類在進行各項生理活動時，神經元都在放電。心臟在跳動時會產生 1～2 毫伏的電壓，眼睛通過開閉就會產生 5～6 毫伏的電壓，然而思考問題時大腦就會產生 0.2～1 毫伏的電壓。BrainLink Lite 正是通過觀察這些腦電波來觀察腦部的活動。

圖 5　Brainlink Lite

2.3.3 設計科學研究法

本次研究採用專門針對資訊系統的設計科學研究法。(如圖 6 所示)

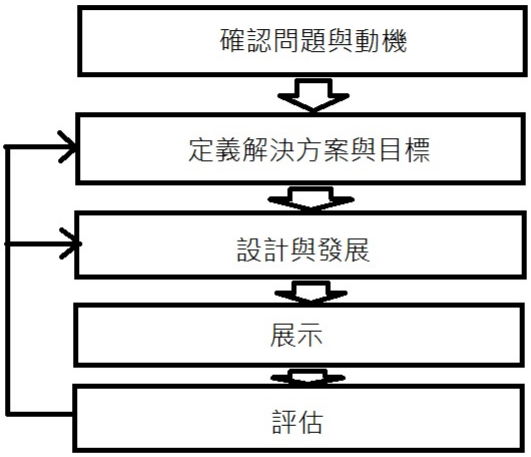


圖 6 設計科學研究法

第一步為確認問題與動機，目的在於將動機與問題明確化，以方便後續的流程。在此步驟必須小心不可將問題以及動機搞錯，否則後續的流程將會受到影響。

第二步為定義解決方案及目標，此步驟須根據上一步所確認的問題提出具體的解決方案以及訂出所要達成的目標。

第三步為設計與發展，根據解決方案來設計系統，並致力於達成所訂定的目標

第四步為展示，將設計完成的系統進行展示及實際運行

第五步為評估，以目標為基準，評估問題的解決程度，以及檢視是否有未解決以及未完成的部分，如果有可以再回到定義解決方案及目標或是設計與發展。

第三章 系統分析與設計

3-1 系統分析與設計

我們採用雛形法中的整體系統雛形模式開發，經需求分析、 設計、編碼 、測試等階段建造一個可運作但不完美的雛型系統。並經由使用者的測試，發掘更完整的需求，使得系統更完整與成熟。

因為我們一開始的想法僅是開發一個 VR 類型棒球遊戲，並想要從中找尋可以延伸出對社會更有貢獻的議題，所以我們需要一個實際能看到的系統雛形，讓使用者試用及收集回饋，了解我們的缺點以及需要增加的功能，使我們的系統更加完整。

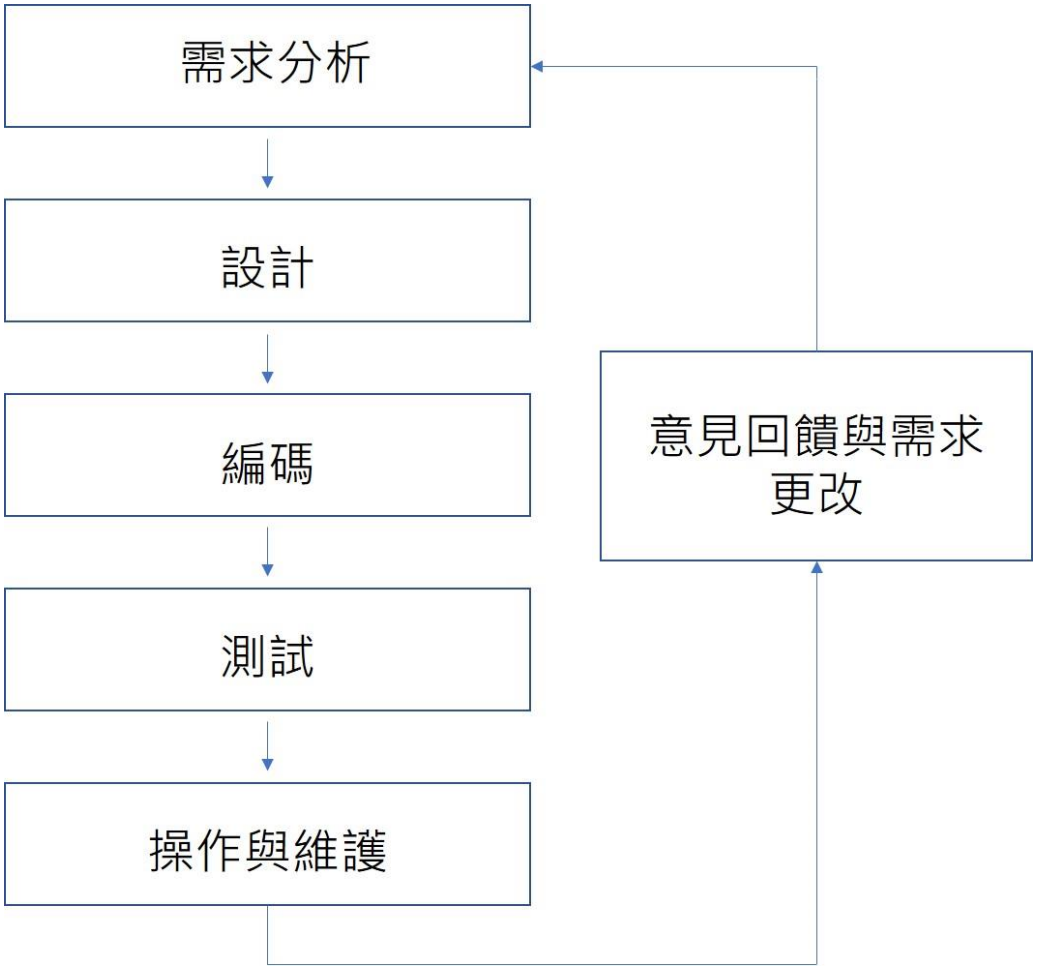


圖 7 系統規劃圖

3-2 系統流程圖

使用者首先帶起 VR 眼鏡與腦波儀，進入遊戲後會先看到使用者介面，並有三種球種可供選擇，移動手把可讓使用者選擇想打擊的球種，選擇完成後就可開始打擊，使用者可隨時用手把上的 MENU 鍵回到使用者介面重新選擇球種,在打擊結束後亦可以返回使用者介面離開遊戲。

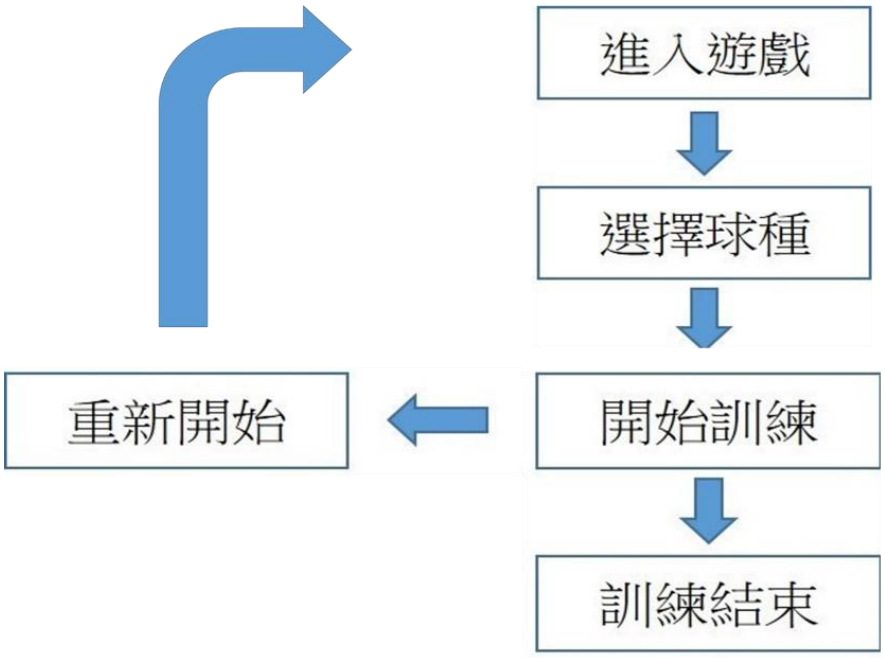


圖 8 系統流程圖

3-3 系統架構圖

使用者帶起 VR 眼鏡與腦波儀，VIVE 控制器以及眼鏡會把資料回傳至串流盒中，串流盒會透過基地台定位使用者的位置，接著透過串流盒至電腦中，並在程式中執行所輸入的動作。

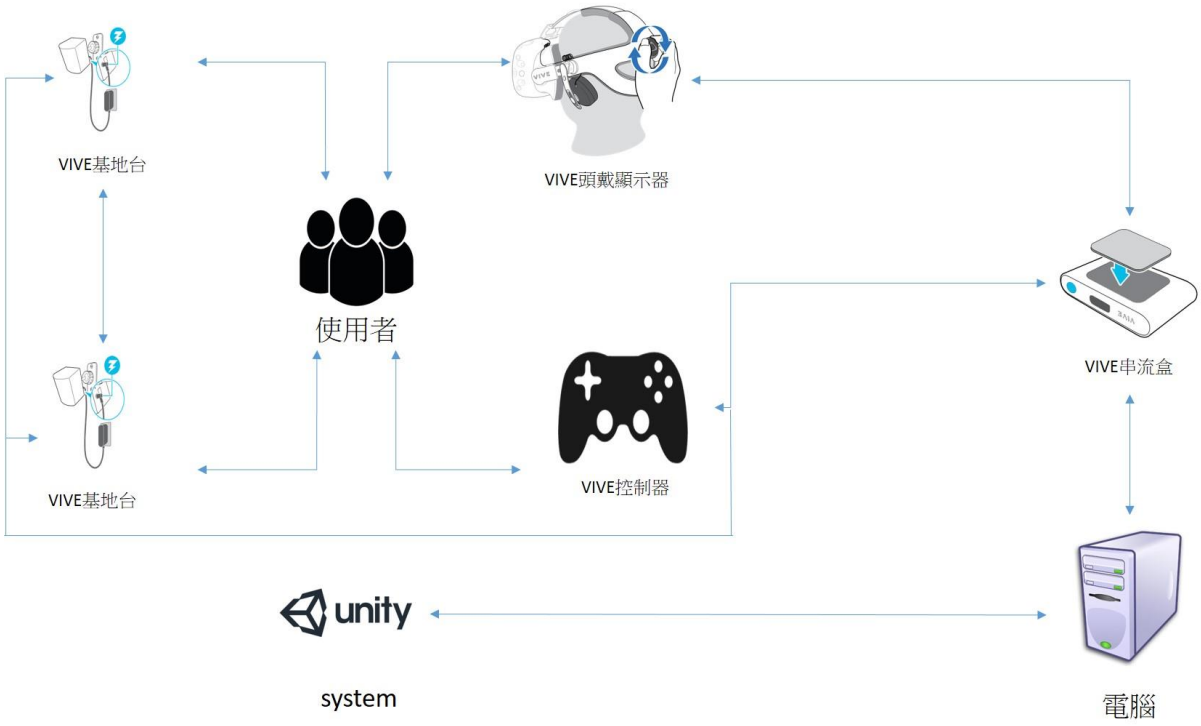


圖 9 系統架構圖

第四章 系統設計與實作

4-1 系統實作

本系統採用 Unity 進行設計，並利用 HTC VIVE 操控。

4-1-1 場景建立

首先建立場景，原本我們是要利用 GOOGLE 街景來製作一個六面體，並將相機放在建立的六面體中，以營造出真實的場景，但這個方法的缺點就是會無法看到投手以及球體完整的飛行過程，因此我們是先建立一個立方體，並將高度調到最低，使立方體形成一個平面，接著將棒球場的平面貼圖附著於平面上，便可建出一個虛擬的棒球場。(如圖 10 所示)

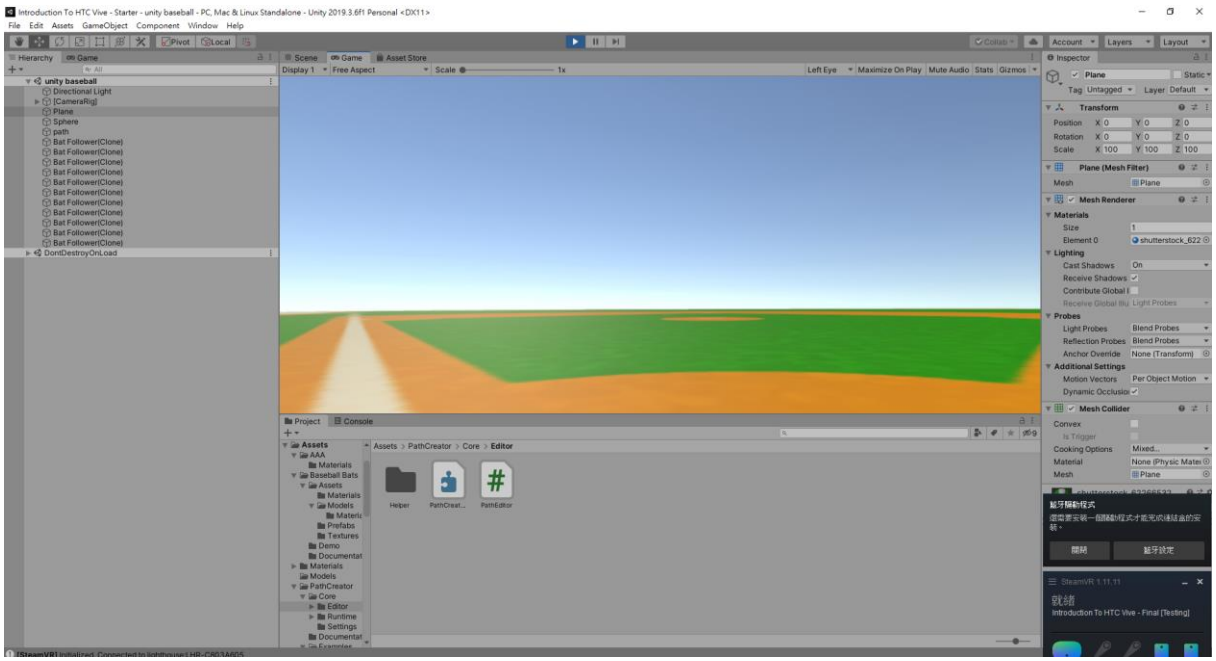


圖 10 完整場景

接下來是操作如何建立場景

首先第一步叫出 3D Object – plane 選擇場景匯入(如圖 11 所示)，

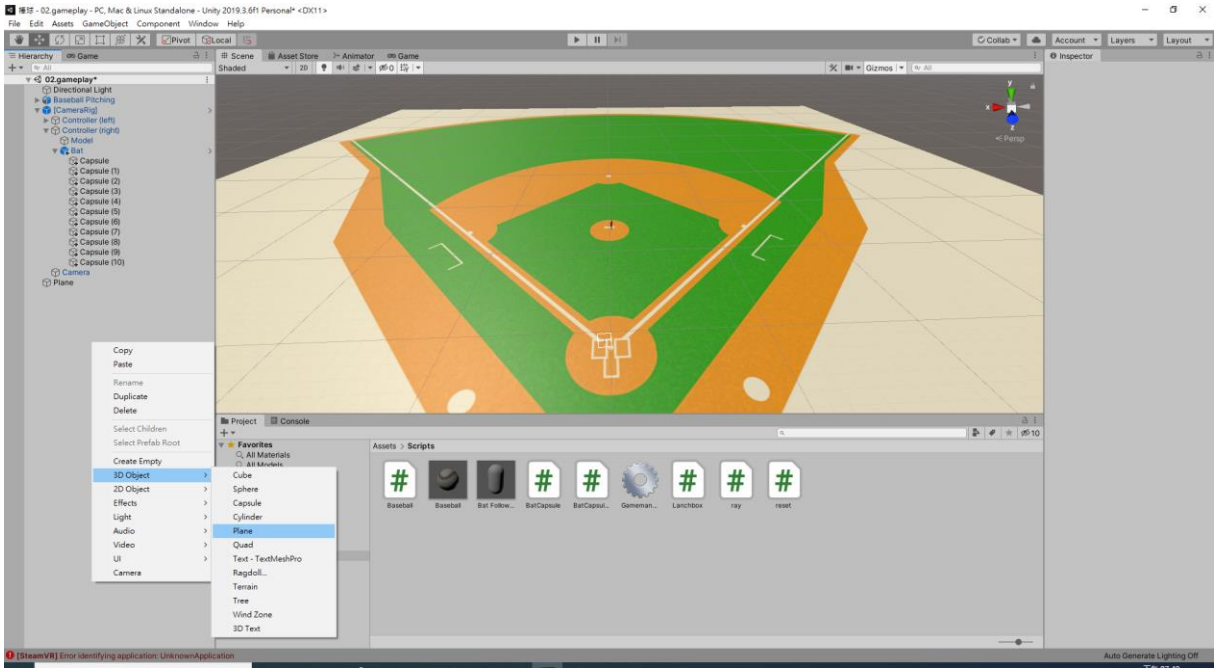


圖 11 場景製作1

調整 Scale 選項 x,y,z 各為 100，使場景成為一整個平面(如圖 12 所示)

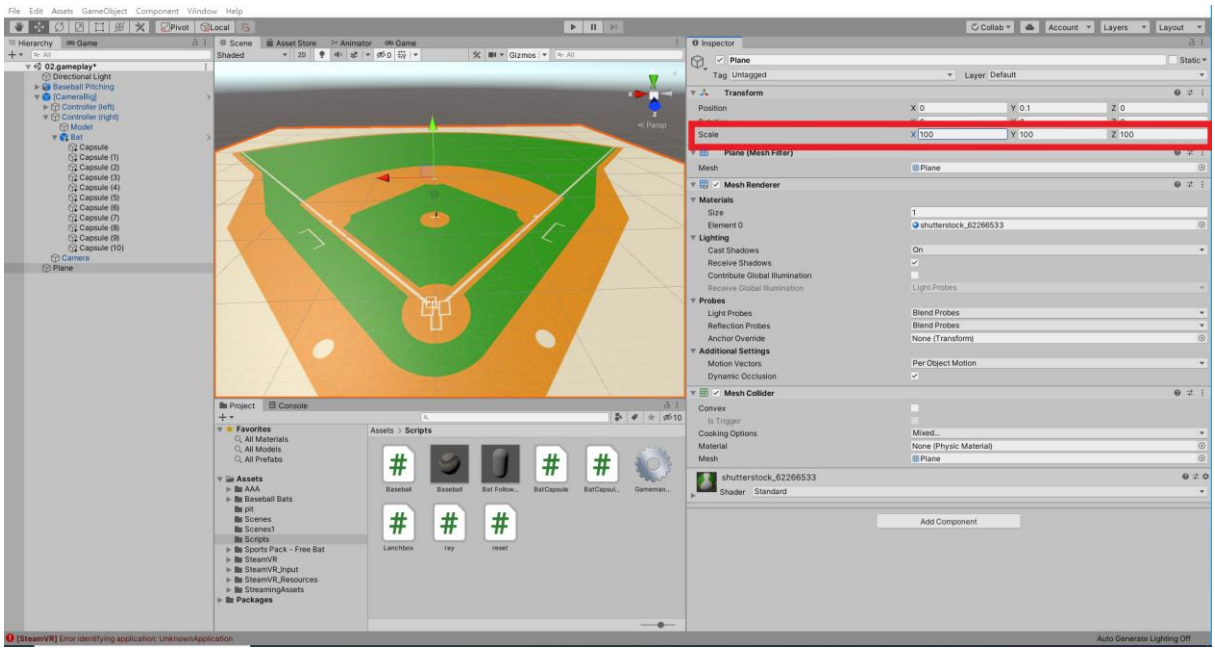


圖 12 場景製作2

匯入場景貼圖(如圖 13 所示)

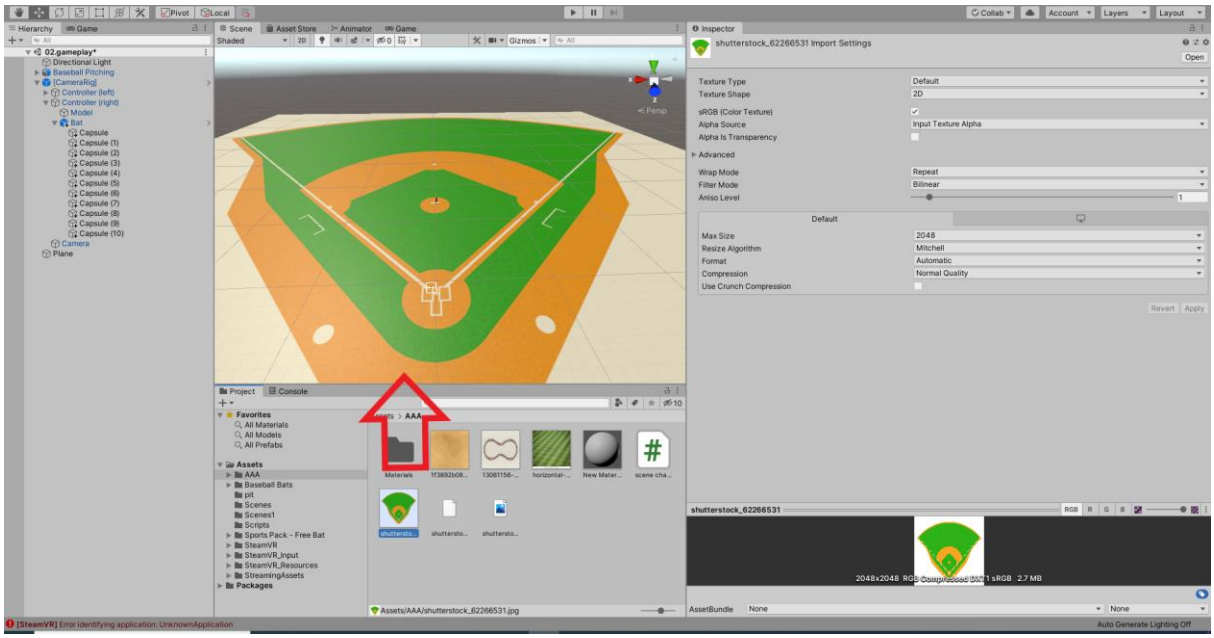


圖 13 場景製作3

4-1-2 球棒

球棒的部分，我們這裡 做了 2 個腳本，分別為 BatCapsule 以及 BatCapsulefollower。BatCapsule 的主要功能是建立數個膠囊體，並將其賦予物理屬性，以便在打擊時能和球體發生物理反應。另一個 BatCapsulefollower 則是將手把，球棒貼圖，以及膠囊體三者做結合。由於球棒貼圖本身不具有任何物理反應，因此需要膠囊體來配合使其成為可以打擊的球棒。而膠囊體則是置入球棒的貼圖中。(如圖 14 所示)

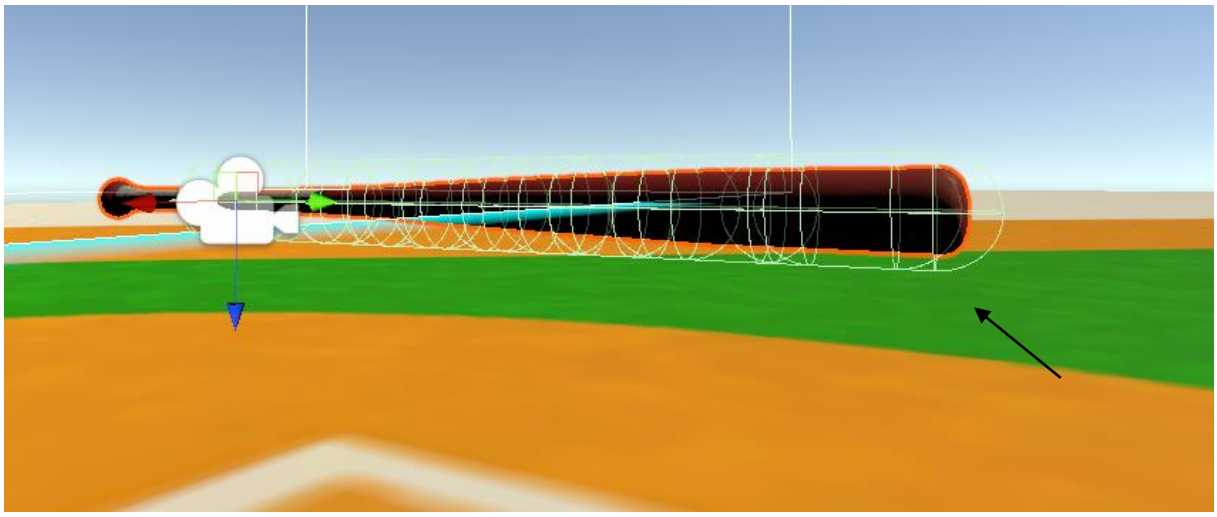
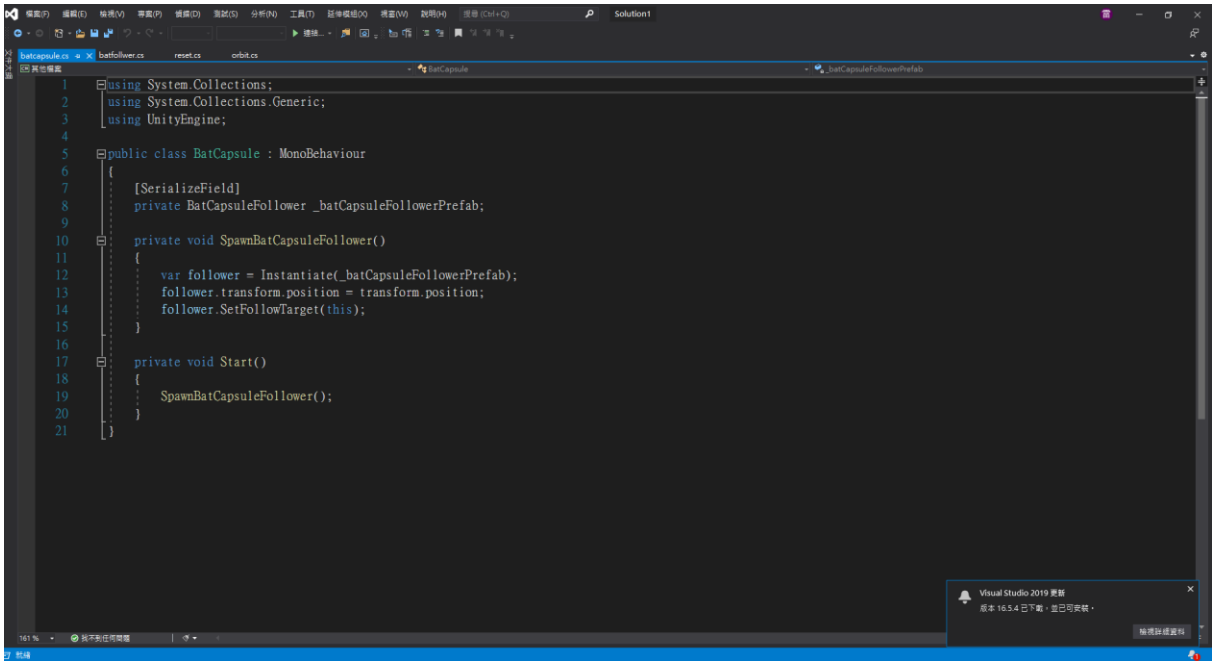


圖 14 球棒示意圖

此程式碼主要功能為呼叫 BatCapsulefollower，使 BatCapsulefollower 可以與 BatCapsule 相互結合在一起。(如圖 15 所示)



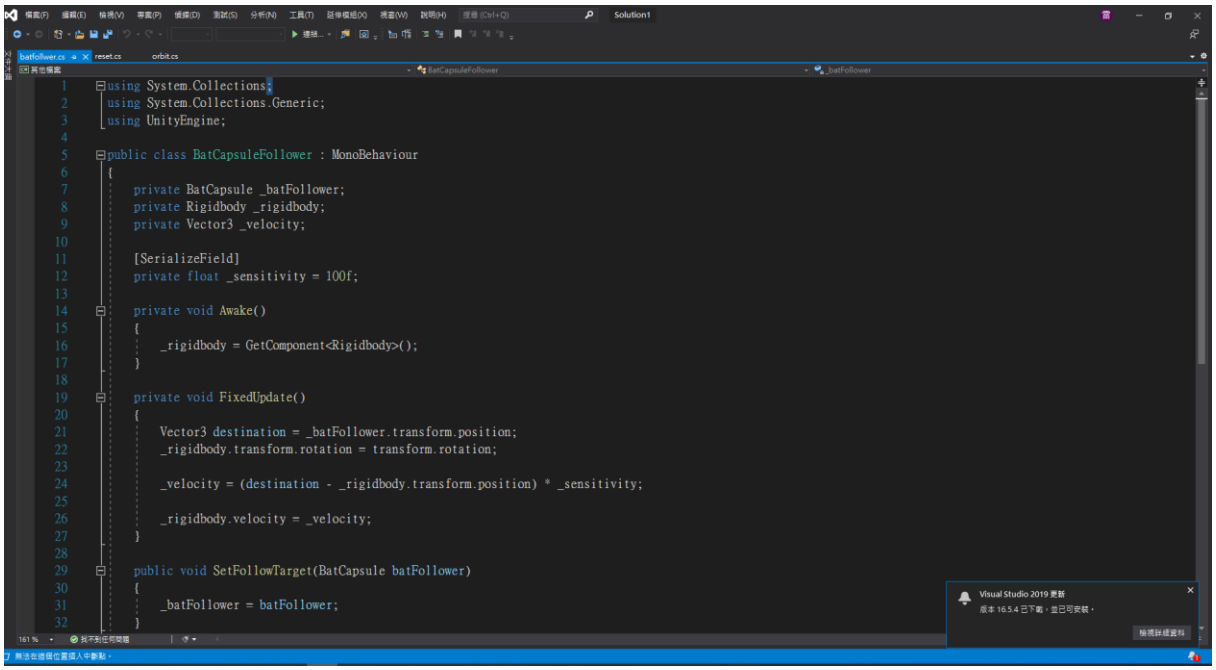
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class BatCapsule : MonoBehaviour
{
    [SerializeField]
    private BatCapsuleFollower _batCapsuleFollowerPrefab;

    private void SpawnBatCapsuleFollower()
    {
        var follower = Instantiate(_batCapsuleFollowerPrefab);
        follower.transform.position = transform.position;
        follower.SetFollowTarget(this);
    }

    private void Start()
    {
        SpawnBatCapsuleFollower();
    }
}
```

圖 15 BatCapsule

遊戲開始執行時會被 BatCapsule 呼叫，同時也會取得 Rigibody(剛體)的屬性。SerializeField 所序列化的則為膠囊附著的敏銳度。FixedUpdate 則是在遊戲執行過程中不斷變化。其中 Vector3 destination 是使球棒可以

依照手把的位置來移動，_rigidbody.transform.rotation 控制的是球棒本體旋轉的方向，

設定錯誤會導致球棒本體的方向錯誤。_velocity 則是基於揮棒的速度，來計算出球與球棒捧狀的數據，最後_rigidbody.velocity 會轉換為球體飛出的速度。(如圖 16 所示)



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class BatCapsuleFollower : MonoBehaviour
{
    private BatCapsule _batFollower;
    private Rigidbody _rigidbody;
    private Vector3 _velocity;

    [SerializeField]
    private float _sensitivity = 100f;

    private void Awake()
    {
        _rigidbody = GetComponent<Rigidbody>();
    }

    private void FixedUpdate()
    {
        Vector3 destination = _batFollower.transform.position;
        _rigidbody.transform.rotation = transform.rotation;

        _velocity = (destination - _rigidbody.transform.position) * _sensitivity;

        _rigidbody.velocity = _velocity;
    }

    public void SetFollowTarget(BatCapsule batFollower)
    {
        _batFollower = batFollower;
    }
```

圖 16 BatCapsulefollower

匯入球棒模型，並把它放入到 Controller(right)底下成為 Controller(right)的子組件(如圖 17 所示)

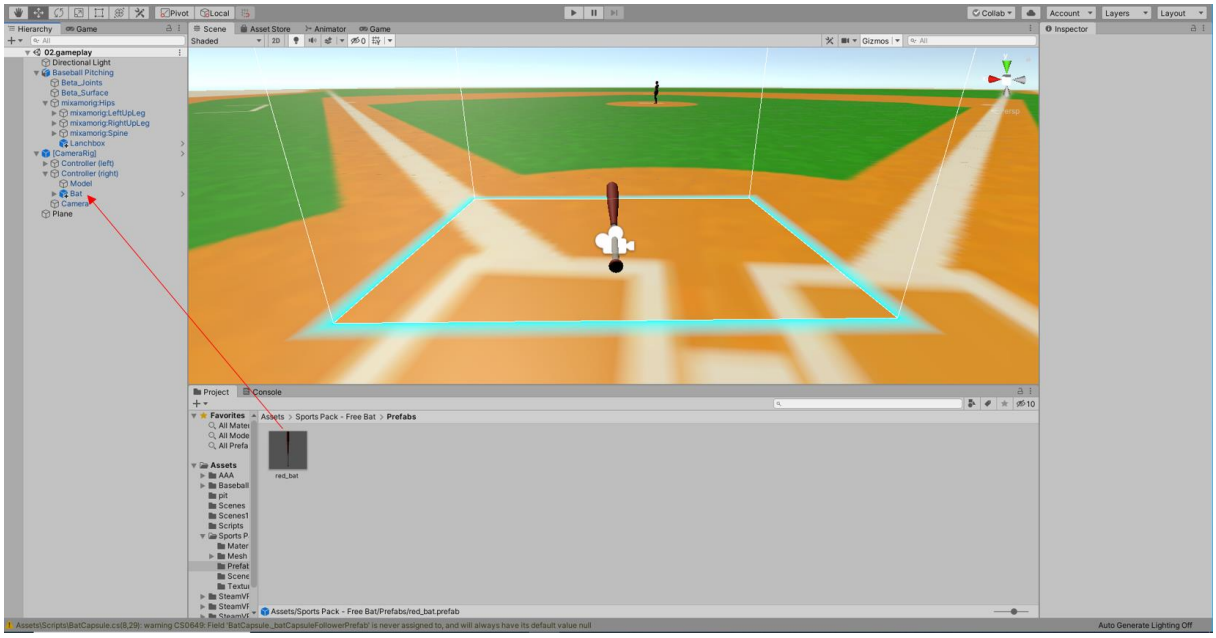


圖 17 球棒製作1

在 Bat 球棒裡加上 Capsule 膠囊體。 (如圖 18 所示)

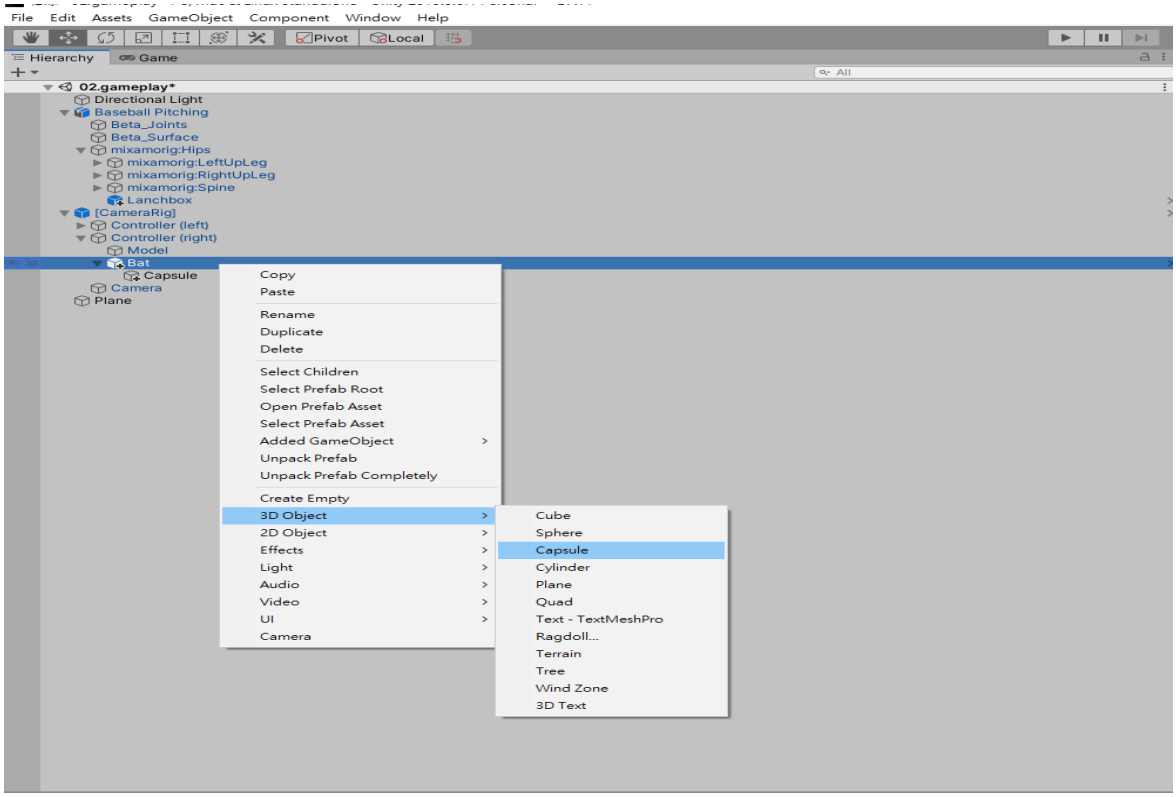


圖 18 球棒製作2

將已經寫好的 BatCapsule 腳本加入 Capsule 中（如圖 19 所示）

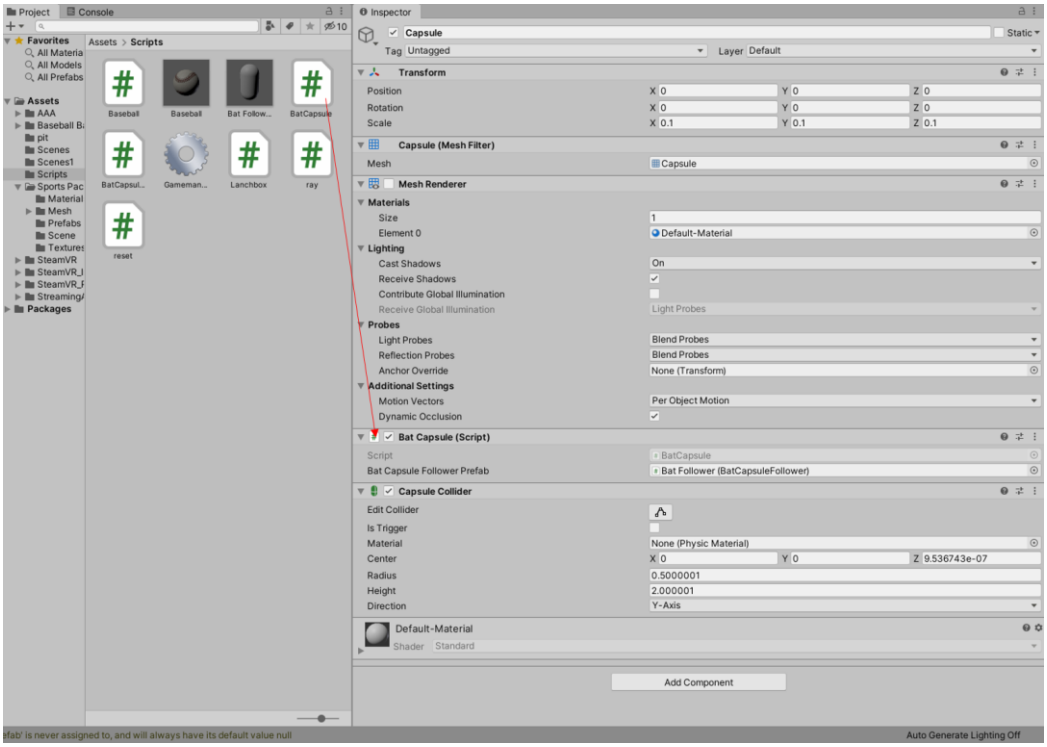


圖 19 球棒製作3

新增一個 Capsule 後把它重新命名為 Bat Follower，再把 BatCapsuleFollower 腳本掛載上去，再拉入 Script 資料夾中製成預製物。(如圖 20 所示)

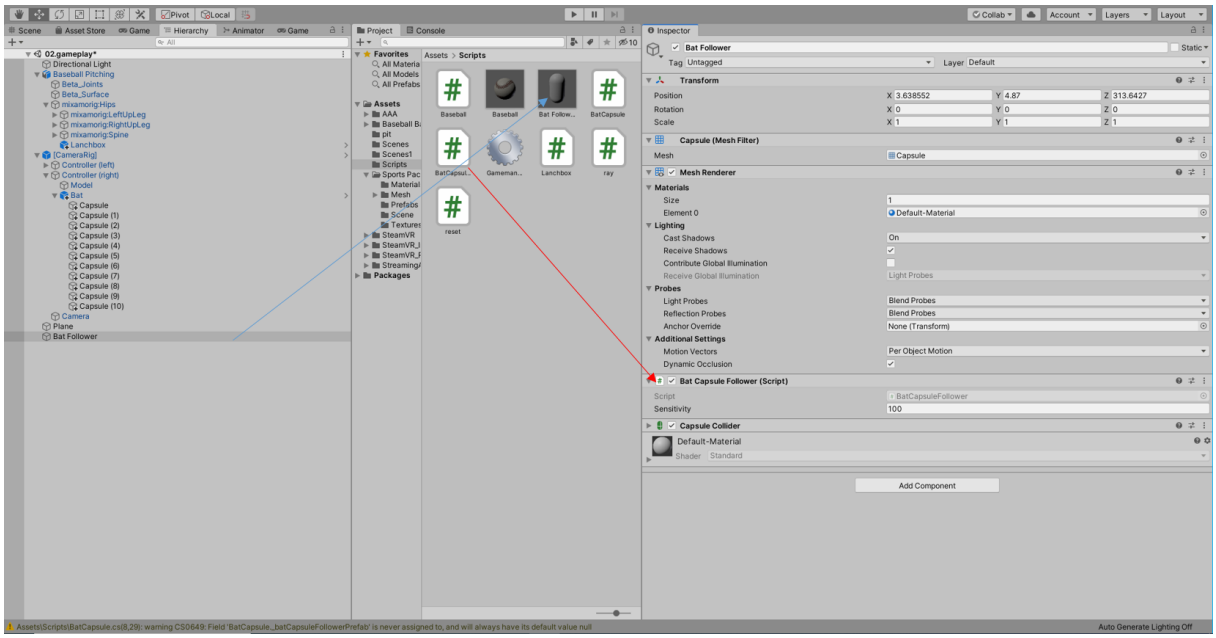


圖 20 球棒製作4

接著再把剛剛製作完的預製物掛載到 Capsule 裡的 BatCapsule 腳本下方的 Bat Capsule Follower Prefab 空格內，就完成了膠囊的製作。再把此膠囊複製數個黏貼在球棒前端就完成了。

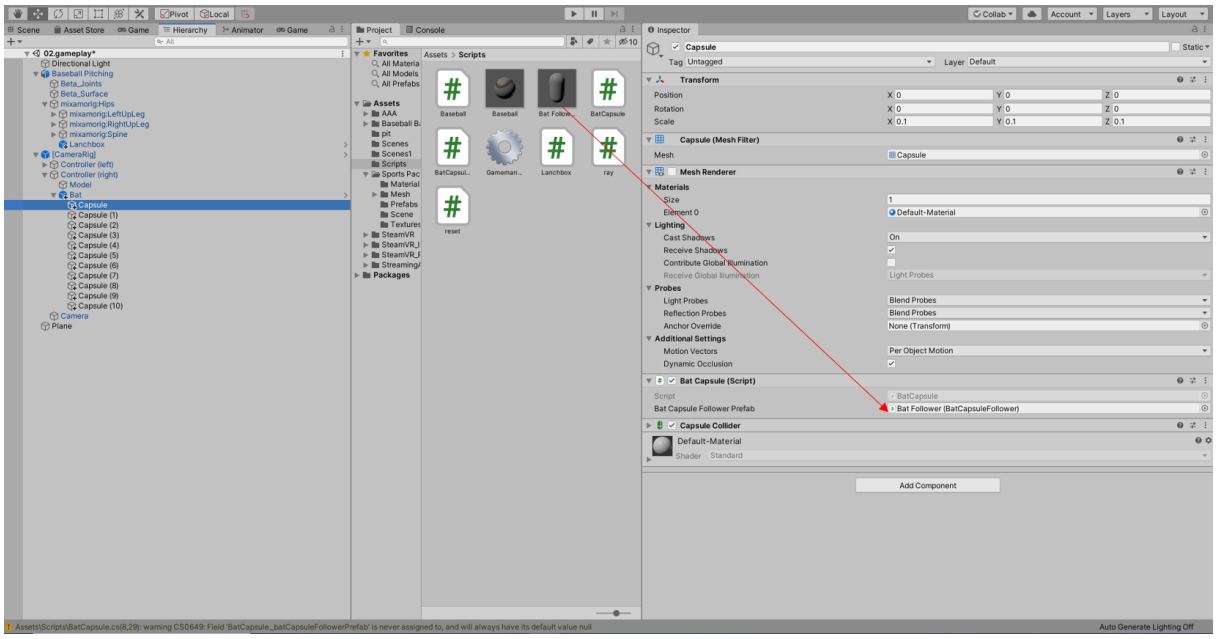


圖 21 球棒製作5

4-1-3 棒球

首先先設定一個球體，接著調整大小再貼上棒球的貼圖，最後將棒球的腳本及材質腳本掛載上這個設定完成的球體，即為我們需要的棒球。(如圖 22 所示)

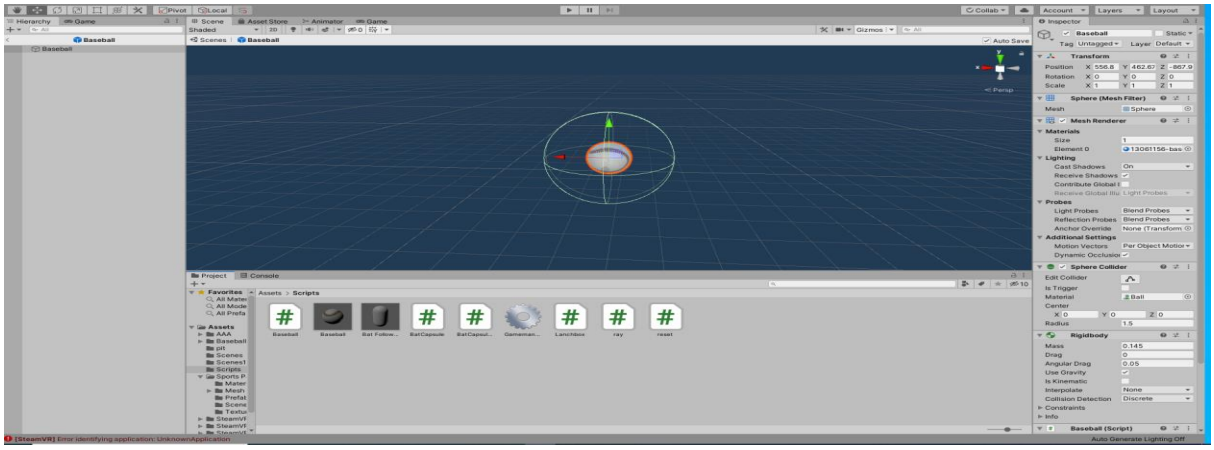


圖 22 完整棒球示意圖

接下來是棒球的實作首先新增一個 Sphere

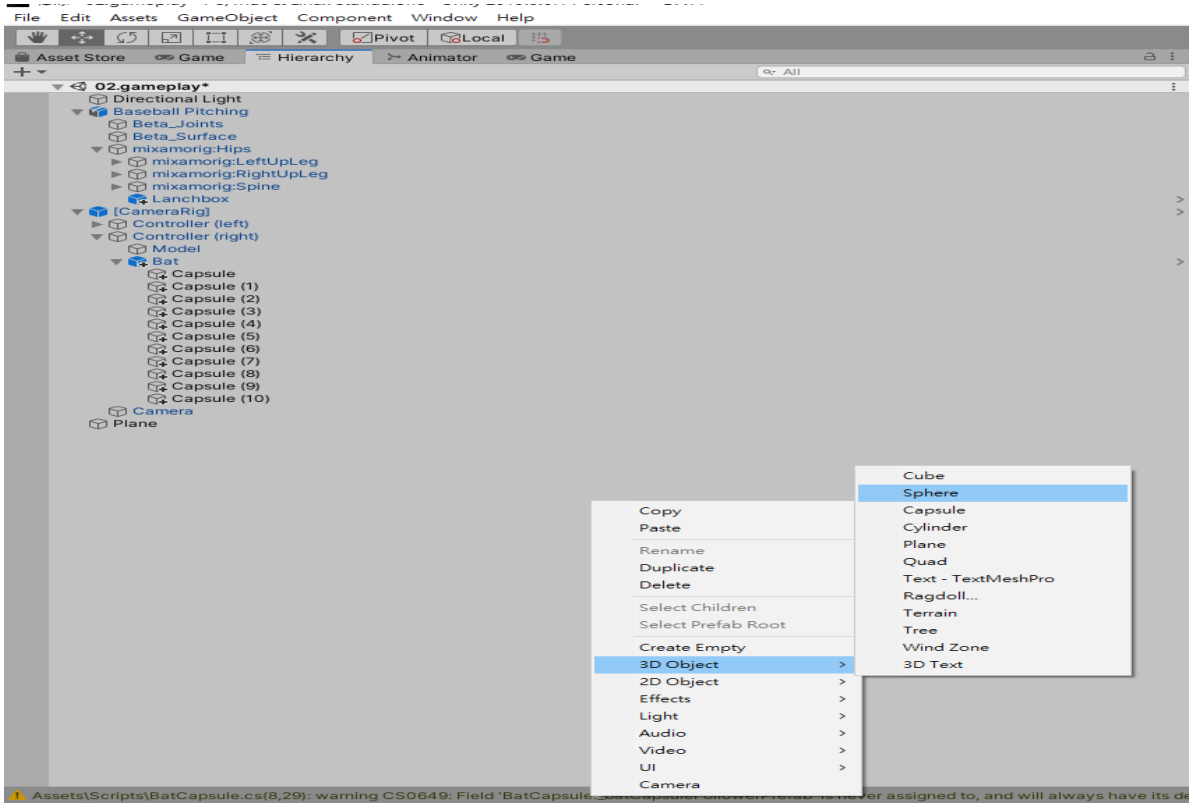


圖 23 棒球製作1

接著改名為 Baseball 再把已經寫好的 Baseball 腳本掛載到此物體上

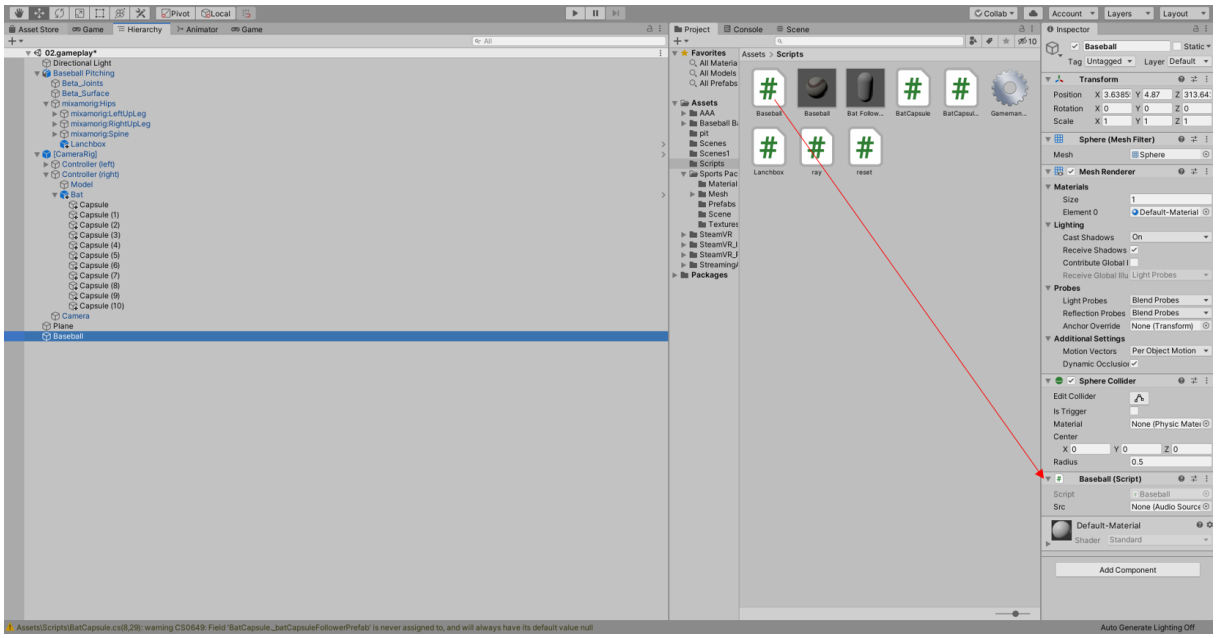


圖 24 棒球製作2

在把棒球貼圖、材質以及 Rigdbody 屬性分別加入在拉入 Script 資料夾中變為預製物就完成了棒球製作。

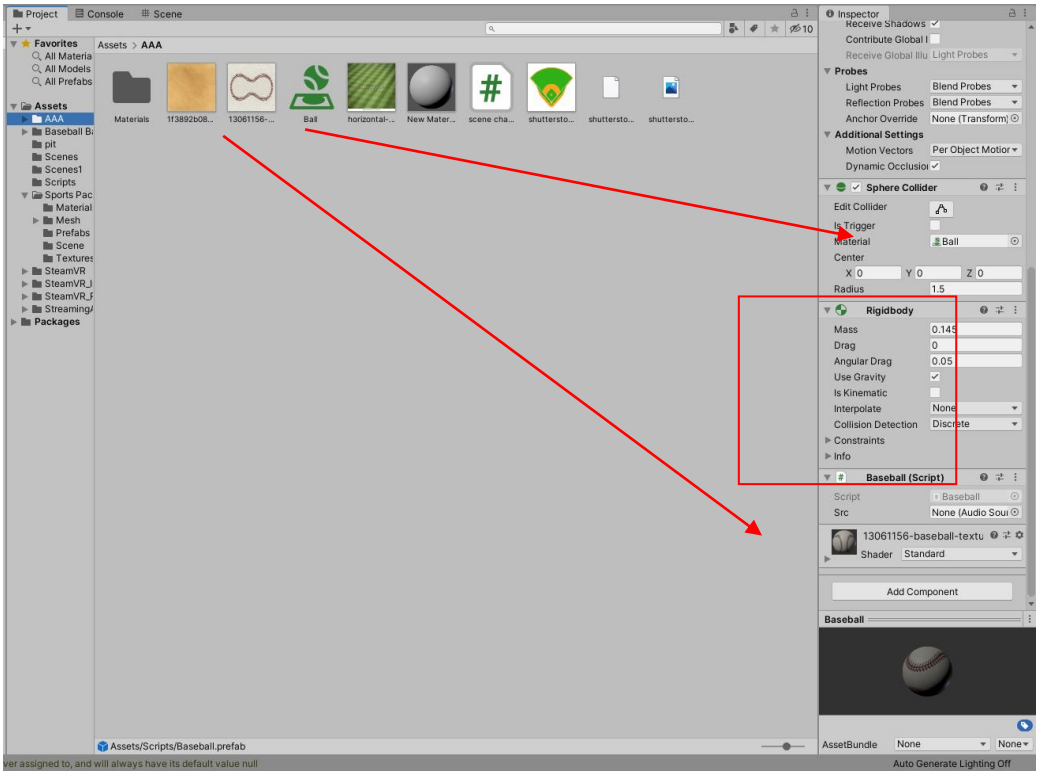


圖 25 棒球製作3

此程式提供球一個初始的力、球速與方向，並判斷球體碰撞到的物體名為 Capsule 時會給予球體一個新的力，並且此力度會因為腦波數值變化的不同而改變，例如：專注度達到 80 且放鬆度達到 60，打擊力度就會放大 50f。(如圖 26 所示)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Valve.VR;

using static BatCapsuleFollower;

0 個參考
public class Baseball : MonoBehaviour
{


    public AudioClip src;
    private AudioSource audioSource;
    public TrailRenderer t;
    DisplayData script = new DisplayData();
    //attention1 = DisplayData.attention1;
    private Rigidbody rb;

    private float velocityMax = 20f;

    0 個參考
    void Awake()
    {
        script = GameObject.Find("NeuroSkyTGCController").GetComponent<DisplayData>();
        rb = gameObject.GetComponent<Rigidbody>();
        rb.AddForce(30, -25, 1700);
        audioSource = GetComponent<AudioSource>();
    }
    1 個參考
    private float GetBatForce(Rigidbody _rigidbody)
    {
        return _rigidbody.velocity.magnitude / velocityMax * 50f;
    }
    1 個參考
    private float GetBatForce2(Rigidbody _rigidbody)
    {
        return _rigidbody.velocity.magnitude / velocityMax * -20f;
    }



        0 個參考
        void OnCollisionEnter(Collision collision)
        {
            if (collision.gameObject.name == "Capsule" && script.attention1 >= 80 && script.meditation1 <= 60)
            {
                rb.velocity = Vector3.zero;
                audioSource.clip = src;
                audioSource.Play();
                Physics.IgnoreCollision(collision.gameObject.GetComponent<CapsuleCollider>(), gameObject.GetComponent<SphereCollider>());


                float forceMultiplier = GetBatForce(collision.gameObject.GetComponent<Rigidbody>());
                Vector3 direction = (transform.position - collision.contacts[0].point).normalized;
                rb.AddForce(direction * forceMultiplier, ForceMode.Impulse);
                rb.useGravity = true;
                Destroy(gameObject, 2f);
            }

            if (collision.gameObject.name == "Capsule" && script.attention1 <= 30 && script.meditation1 >= 60)
            {
                rb.velocity = Vector3.zero;
                audioSource.clip = src;
                audioSource.Play();
                Physics.IgnoreCollision(collision.gameObject.GetComponent<CapsuleCollider>(), gameObject.GetComponent<SphereCollider>());


                float forceMultiplier = GetBatForce2(collision.gameObject.GetComponent<Rigidbody>());
                Vector3 direction = (transform.position - collision.contacts[0].point).normalized;
                rb.AddForce(direction * forceMultiplier, ForceMode.Impulse);
                rb.useGravity = true;
                Destroy(gameObject, 2f);
            }
```

圖 26 直球製作

Constant Force，此一元件是遊戲開始後持續對球體 X 軸 Y 軸 Z 軸給予一種方向力度，以直球製作為基礎，調整發射角度並持續給予球一個向下的力，即可完成曲球。(如圖 27 所示)

圖 27 曲球製作

同樣利用 Constant Force，以直球製作為基礎，調整發射角度並持續給予球一個向左的力，即可完成滑球。(如圖 28 所示)

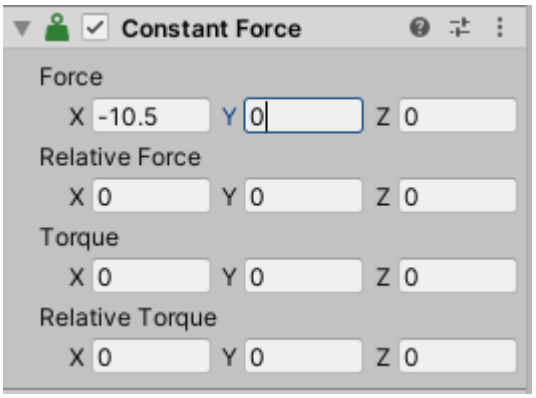


圖 28 滑球製作

4-1-4 物理碰撞

構成物理碰撞的條件需要碰撞雙方的物體均需加上 Rigidbody 屬性，並且其中一方需另外加上 Collider 屬性，因此我們將球與球棒均加上 Rigidbody，再將球棒加上 Collider，達成物理碰撞。

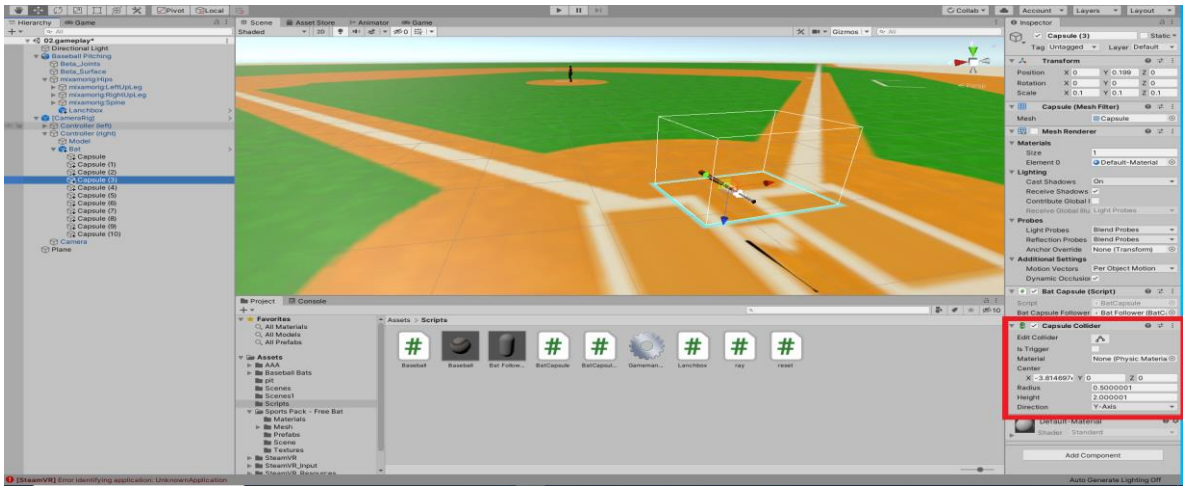


圖 29 膠囊體剛體

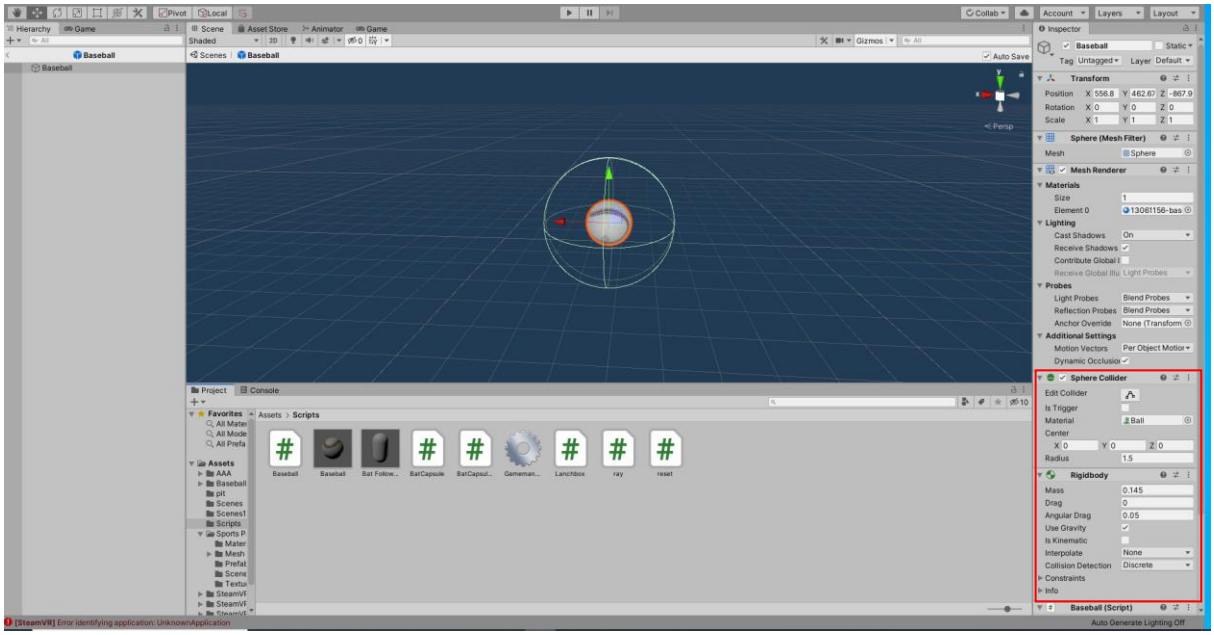


圖 30 棒球碰撞器與剛體

4-1-5 投手動畫

投手的動畫我們是設定循環播放，並配合 LaunchBox 的發球時間，達成投手投球的目標。

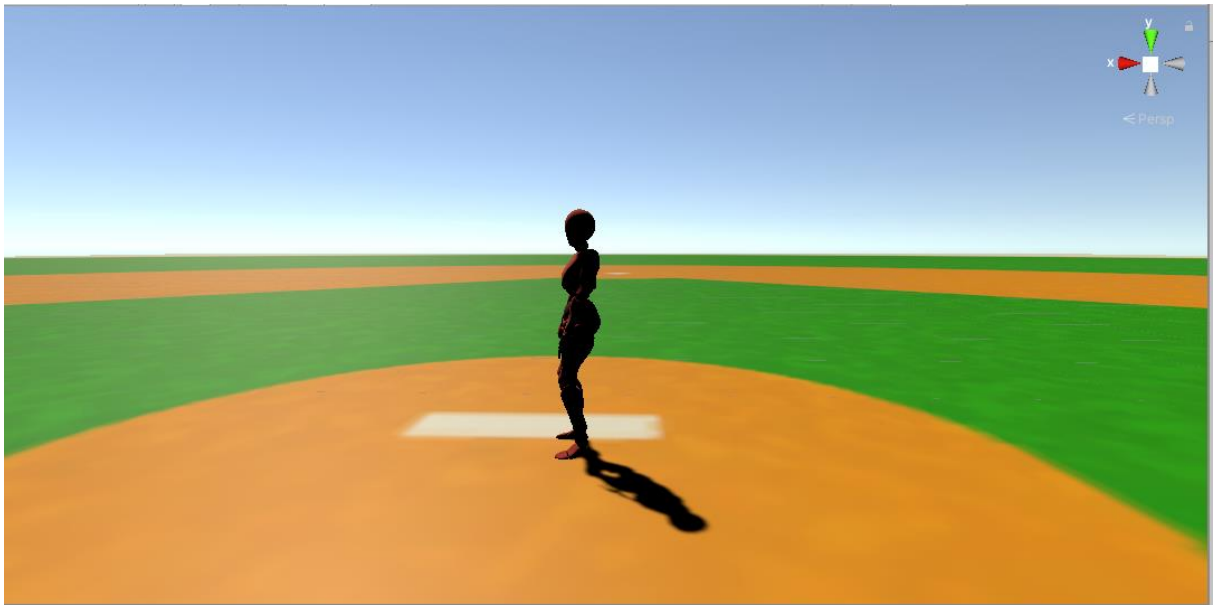


圖 31 投手動畫示意圖

4-1-6 動畫控制

首先要先建立一個動畫控制器。下圖中紅色圈起來的部分為開始，接著開始播放投手的動畫，紅色的 EXIT 表示動畫結束。在投手動畫設定頁面可以設定循環播放以及投手的動畫時間。(如圖 32 中紅色圓圈所示)

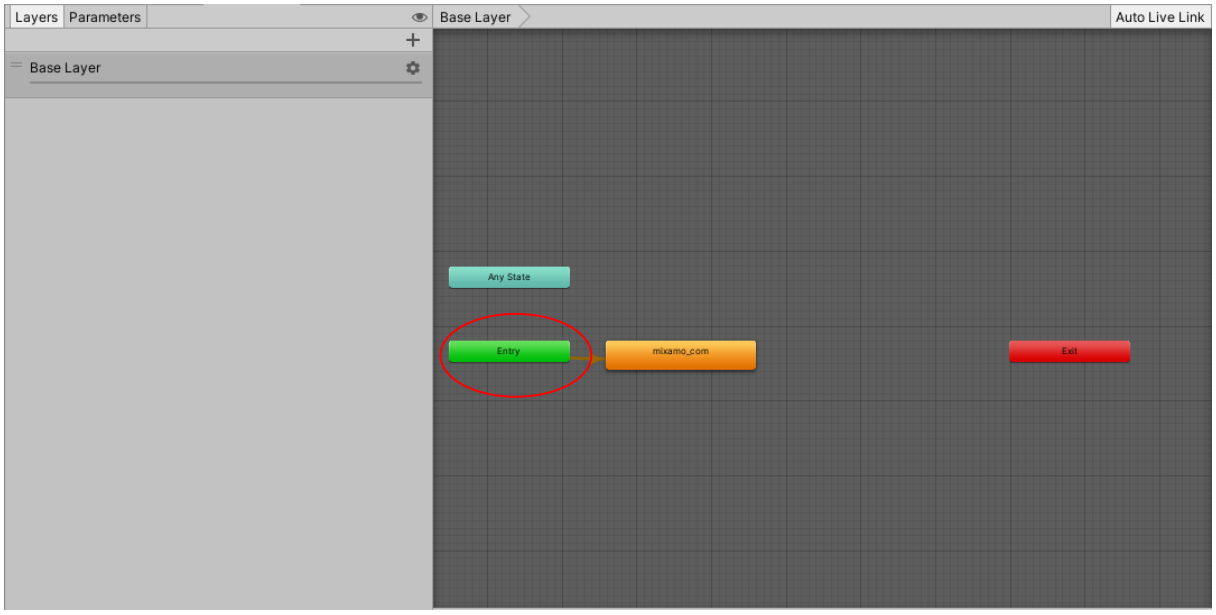


圖 32 動畫控制

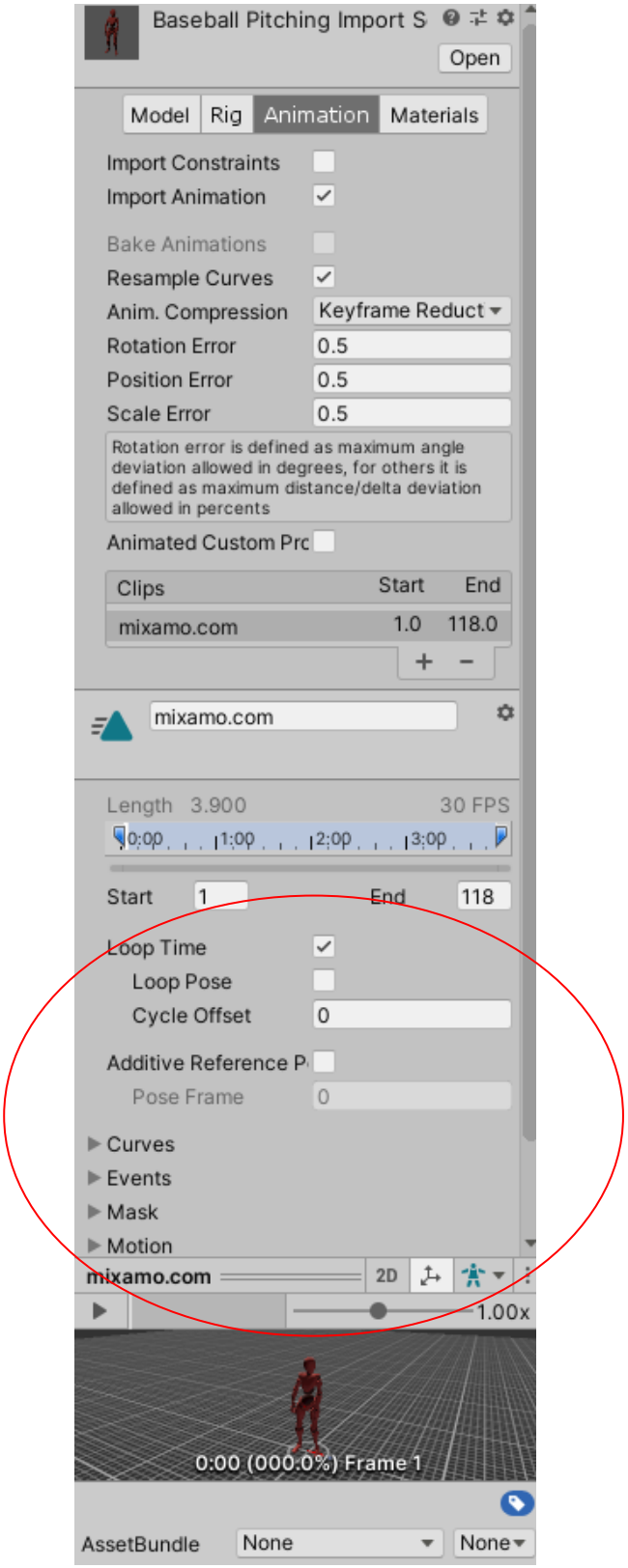

Baseball Pitching Import S
Open
Model
Rig
Animation
Materials
Import Constraints
Import Animation
Bake Animations
Resample Curves
Anim. Compression
Keyframe Reduct
Rotation Error
0.5
Position Error
0.5
Scale Error
0.5
Rotation error is defined as maximum angle deviation allowed in degrees, for others it is defined as maximum distance/delta deviation allowed in percents
Animated Custom Prc
Clips
Start
End
mixamo.com
1.0
118.0
mixamo.com
Length 3.900
30 FPS
0:00
1:00
2:00
3:00
Start
1
End
118
Loop Time
Loop Pose
Cycle Offset
0
Additive Reference P
Pose Frame
0
Curves
Events
Mask
Motion
mixamo.com
2D
1.00x
0:00 (000.0%) Frame 1
AssetBundle
None
None

圖 33 動畫控制2

4-1-7 發球點

發球點掛載著 Launchbox 腳本，並放置在投手動畫前，能與投手動畫搭配。

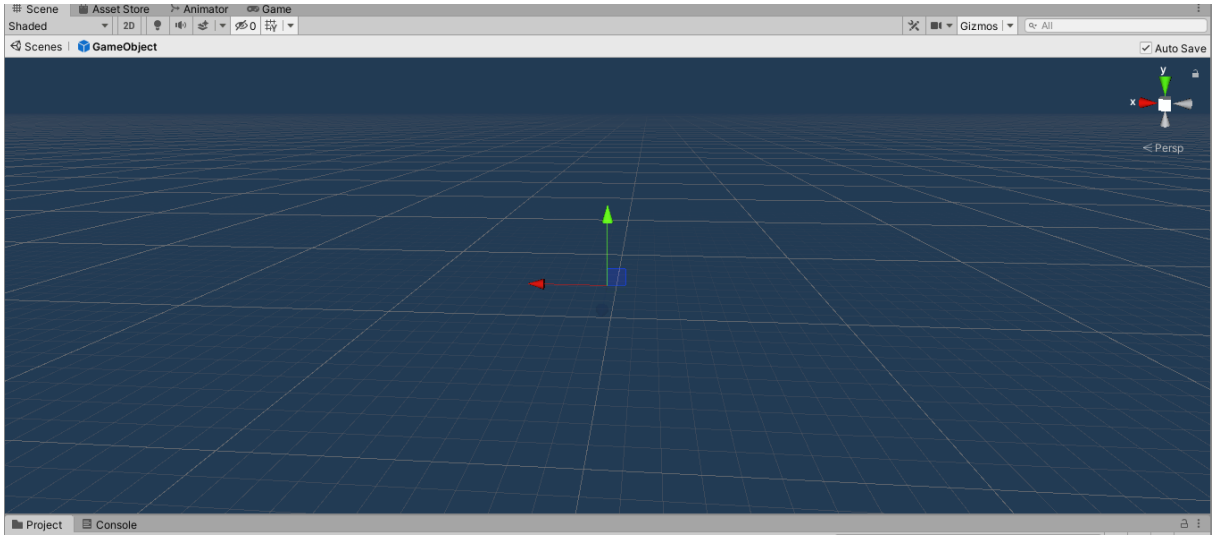


圖 34 發球點

首先宣告第一個類別 Launchbox，接著在 Launchbox 下宣告三個 public 函數，分別為投球間隔時間，遊戲物件，以及發球角度。

宣告 private 函數，這裡使用計數器 IEnumerator，在程式中會先取得 GetLaunchDirection 所產生的隨機數值，並經由 Quaternion 函數的運算，產生球投出的角度，速度並投出

GetLaunchDirection 則為 private 函數，會隨機產生仰角的數值，並回傳給 Pitch 函數。

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Lanchbox : MonoBehaviour
{
    public float timeBetweenPitches;
    public GameObject ball;
    public float launchAngle;

    void Start()
    {
        StartCoroutine(Pitch());
    }

    private IEnumerator Pitch()
    {
        while (true)
        {
            yield return new WaitForSeconds(timeBetweenPitches);
            Vector3 launchDirection = GetLaunchDirection();
            Quaternion q = Quaternion.Euler(launchDirection);

            Instantiate(ball, transform.position, q);
        }
    }

    private Vector3 GetLaunchDirection()
    {
        return new Vector3(
            launchAngle, 0, Random.Range(0f, 180f));
    }
}

```

圖 35 Launchbox程式碼

4-1-8 使用者介面

此介面是使用者進入遊戲時可以選擇的選單，功能為選擇球種與結束遊戲。使用者能自行選擇自己想打擊的球種，選擇完成後執行開始遊戲會自動跳轉遊戲場景。

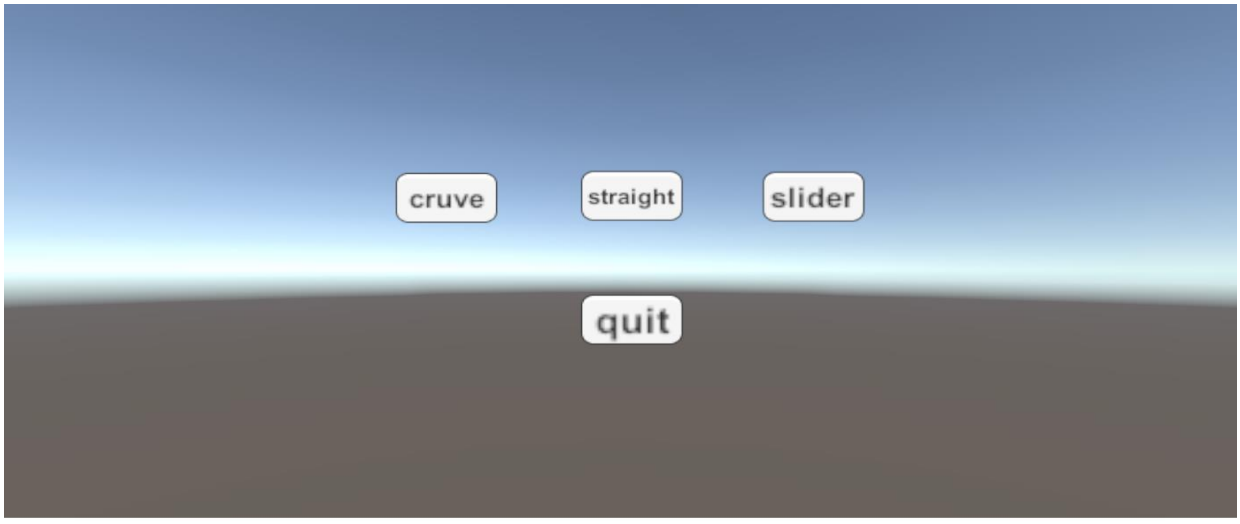


圖 36 使用者介面

4-1-9 腦波

藉由程式碼(如圖 37、圖 38 所示) 將腦波設備連接。

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;
using Jayrock.Json;
using Jayrock.Json.Conversion;
using System.Net.Sockets;
using System.Text;
using System.IO;

2 個參考
public class TGCConnectionController : MonoBehaviour {
    private TcpClient client;
    private Stream stream;
    private byte[] buffer;

    public delegate void UpdateIntValueDelegate(int value);
    public delegate void UpdateFloatValueDelegate(float value);

    public event UpdateIntValueDelegate UpdatePoorSignalEvent;
    public event UpdateIntValueDelegate UpdateAttentionEvent;
    public event UpdateIntValueDelegate UpdateMeditationEvent;
    public event UpdateIntValueDelegate UpdateRawdataEvent;
    public event UpdateIntValueDelegate UpdateBlinkEvent;

    public event UpdateFloatValueDelegate UpdateDeltaEvent;
    public event UpdateFloatValueDelegate UpdateThetaEvent;
    public event UpdateFloatValueDelegate UpdateLowAlphaEvent;
    public event UpdateFloatValueDelegate UpdateHighAlphaEvent;
    public event UpdateFloatValueDelegate UpdateLowBetaEvent;
    public event UpdateFloatValueDelegate UpdateHighBetaEvent;
    public event UpdateFloatValueDelegate UpdateLowGammaEvent;
    public event UpdateFloatValueDelegate UpdateHighGammaEvent;


    0 個參考
    void Start () {
        //Connect();
    }

    2 個參考
    public void Disconnect(){
        if(IsInvoking("ParseData")){
            CancelInvoke("ParseData");
            stream.Close();
        }
    }

    1 個參考
    public void Connect(){
        if(!IsInvoking("ParseData")){

            client = new TcpClient("127.0.0.1", 13854);
            stream = client.GetStream();
            buffer = new byte[1024];
            byte[] myWriteBuffer = Encoding.ASCII.GetBytes(@"{""enableRawOutput"": true, ""format"": ""Json""}");
            stream.Write(myWriteBuffer, 0, myWriteBuffer.Length);

            InvokeRepeating("ParseData",0.1f,0.12f);
        }
    }
    0 個參考
    IEnumerator abc()
    {
        if (!IsInvoking("ParseData"))
        {

            client = new TcpClient("127.0.0.1", 13854);
            stream = client.GetStream();
            buffer = new byte[1024];
            byte[] myWriteBuffer = Encoding.ASCII.GetBytes(@"{""enableRawOutput"": true, ""format"": ""Json""}");
            stream.Write(myWriteBuffer, 0, myWriteBuffer.Length);

            InvokeRepeating("ParseData", 0.1f, 0.02f);
        }
```

圖 37 腦波設備連接1

```
        yield return new WaitForSeconds(0.5f);
    }

    0 個參考
    void ParseData(){
        if(stream.CanRead){
          try {
            int bytesRead = stream.Read(buffer, 0, buffer.Length);

            string[] packets = Encoding.ASCII.GetString(buffer, 0, bytesRead).Split('\r');

            foreach(string packet in packets){
              if(packet.Length == 0)
                continue;

              IDictionary primary = (IDictionary)JsonConvert.Import(typeof(IDictionary), packet);

              if(primary.Contains("poorSignalLevel")){

                if(UpdatePoorSignalEvent != null){
                  UpdatePoorSignalEvent(int.Parse(primary["poorSignalLevel"].ToString()));
                  }

                if(primary.Contains("eSense")){
                  IDictionary eSense = (IDictionary)primary["eSense"];
                  if(UpdateAttentionEvent != null){
                    UpdateAttentionEvent(int.Parse(eSense["attention"].ToString()));
                    }
                  if(UpdateMeditationEvent != null){
                    UpdateMeditationEvent(int.Parse(eSense["meditation"].ToString()));
                    }
                  }

                if(primary.Contains("eegPower")){
                  IDictionary eegPowers = (IDictionary)primary["eegPower"];

                  if(UpdateDeltaEvent != null){
                    UpdateDeltaEvent(float.Parse(eegPowers["delta"].ToString()));
                    }
                  if(UpdateThetaEvent != null){
                    UpdateThetaEvent(float.Parse(eegPowers["theta"].ToString()));
                    }
                  if(UpdateLowAlphaEvent != null){
                    UpdateLowAlphaEvent(float.Parse(eegPowers["lowAlpha"].ToString()));
                    }
                  if(UpdateHighAlphaEvent != null){
                    UpdateHighAlphaEvent(float.Parse(eegPowers["highAlpha"].ToString()));
                    }
                  if(UpdateLowBetaEvent != null){
                    UpdateLowBetaEvent(float.Parse(eegPowers["lowBeta"].ToString()));
                    }
                  if(UpdateHighBetaEvent != null){
                    UpdateHighBetaEvent(float.Parse(eegPowers["highBeta"].ToString()));
                    }
                  if(UpdateLowGammaEvent != null){
                    UpdateLowGammaEvent(float.Parse(eegPowers["lowGamma"].ToString()));
                    }
                  if(UpdateHighGammaEvent != null){
                    UpdateHighGammaEvent(float.Parse(eegPowers["highGamma"].ToString()));
                    }
                  }
                }
                else if(primary.Contains("rawEeg") && UpdateRawdataEvent != null){
                      UpdateRawdataEvent(int.Parse(primary["rawEeg"].ToString()));
                }
                else if(primary.Contains("blinkStrength") && UpdateBlinkEvent != null){
                      UpdateBlinkEvent(int.Parse(primary["blinkStrength"].ToString()));
                }
              }
            }
            catch(IOException e){ Debug.Log("IOException " + e); }
            catch(System.Exception e){ Debug.Log("Exception " + e); }
        }
```

圖 38 腦波設備連接2

連接後藉由程式碼(如圖39所示)收集數據並將數據投放在螢幕左上角。(如圖40所示)

```
using UnityEngine;
using System.Collections;


9 個參考
public class DisplayData : MonoBehaviour
{
    public Texture2D[] signalIcons;

    private int indexSignalIcons = 1;

    TGCConnectionController controller;

    private int poorSignal;
    public int attentional;
    public int meditational;

    private float delta;

    0 個參考
    void Start()
    {

        controller = GameObject.Find("NeuroSkyTGCController").GetComponent<TGCConnectionController>();

        controller.UpdatePoorSignalEvent += OnUpdatePoorSignal;
        controller.UpdateAttentionEvent += OnUpdateAttention;
        controller.UpdateMeditationEvent += OnUpdateMeditation;

        controller.UpdateDeltaEvent += OnUpdateDelta;


    }

    1 個參考
    void OnUpdatePoorSignal(int value){
        poorSignal = value;
        if(value < 25){
            indexSignalIcons = 0;
        }else if(value >= 25 && value < 51){
            indexSignalIcons = 4;
        }else if(value >= 51 && value < 78){
            indexSignalIcons = 3;
        }else if(value >= 78 && value < 107){
            indexSignalIcons = 2;
        }else if(value >= 107){
            indexSignalIcons = 1;
        }
    }
    1 個參考
    void OnUpdateAttention(int value){
        attentional = value;
    }
    1 個參考
    void OnUpdateMeditation(int value){
        meditational = value;
    }
    1 個參考
    void OnUpdateDelta(float value){
        delta = value;
    }


    0 個參考
    void OnGUI()
    {
        GUILayout.BeginHorizontal();


        if (GUILayout.Button("Connect"))
        {
            controller.Connect();
        }
        if (GUILayout.Button("DisConnect"))
        {
            controller.Disconnect();
            indexSignalIcons = 1;
        }

        GUILayout.Space(Screen.width-250);
        GUILayout.Label(signalIcons[indexSignalIcons]);

        GUILayout.EndHorizontal();



            GUILayout.Label("PoorSignal:" + poorSignal);
            GUILayout.Label("Attention:" + attentional);
            GUILayout.Label("Meditational:" + meditational);
            GUILayout.Label("Delta:" + delta);

    }
}
```

圖 39 .收集數據及展示

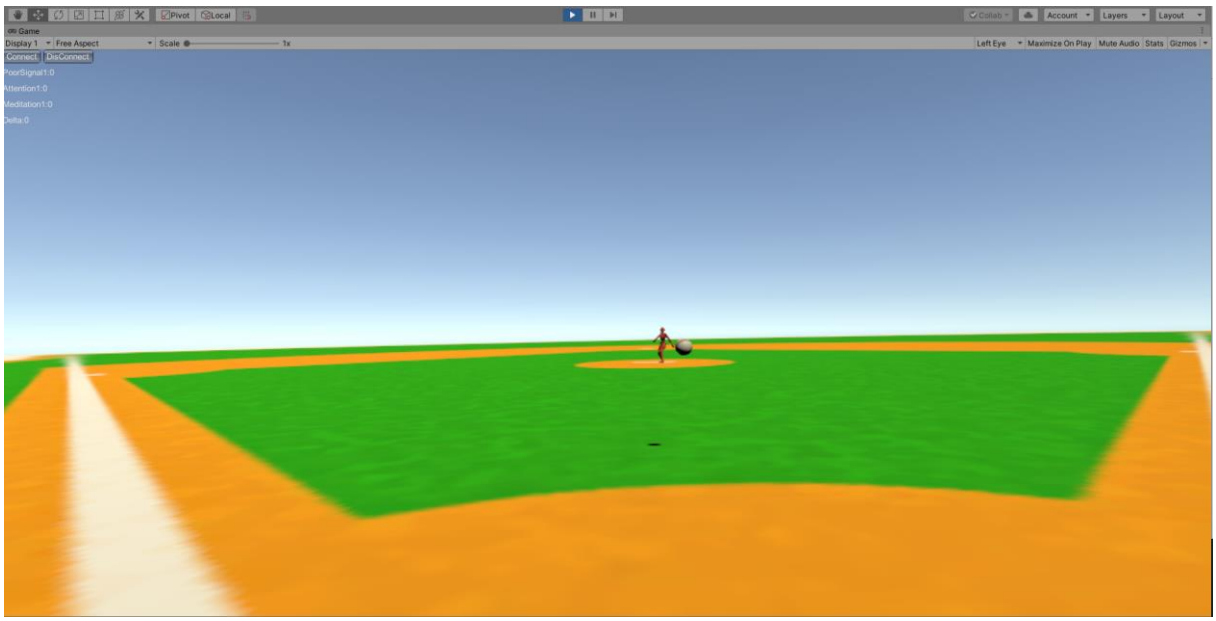

Pivot
Local
Account
Layers
Layout
Game
Display 1
Free Aspect
Scale
1x
Left Eye
Maximize On Play
Mute Audio
Stats
Gizmos
Connect
DisConnect
PoorSignal1:0
Attention1:0
Meditation1:0
Delta:0

圖 40 腦波數據展示

第五章 系統展示

5-1 系統展示

找出棒球遊戲程式，並開始執行。

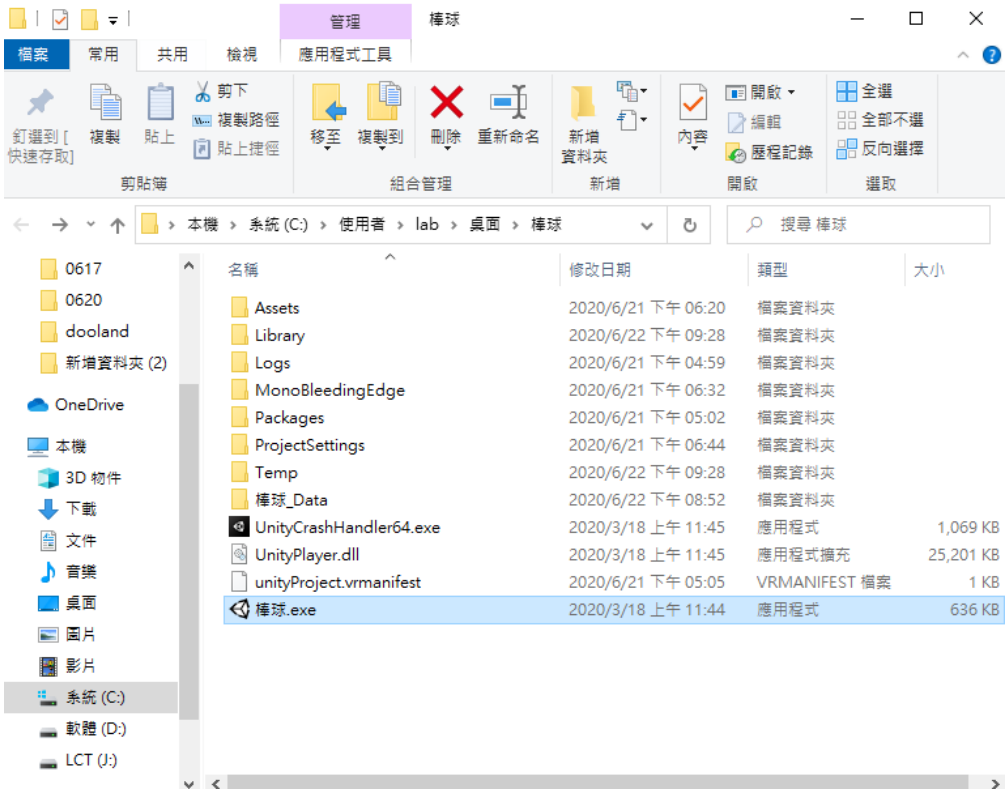


圖 41 遊戲程式資料夾

圖 42 遊戲開啟畫面

配戴腦波設備及 VR 設備後即即可開始遊戲。

打開程式後會跳到使用者介面,此畫面使用者可以選擇想打擊的球種與結束遊戲。

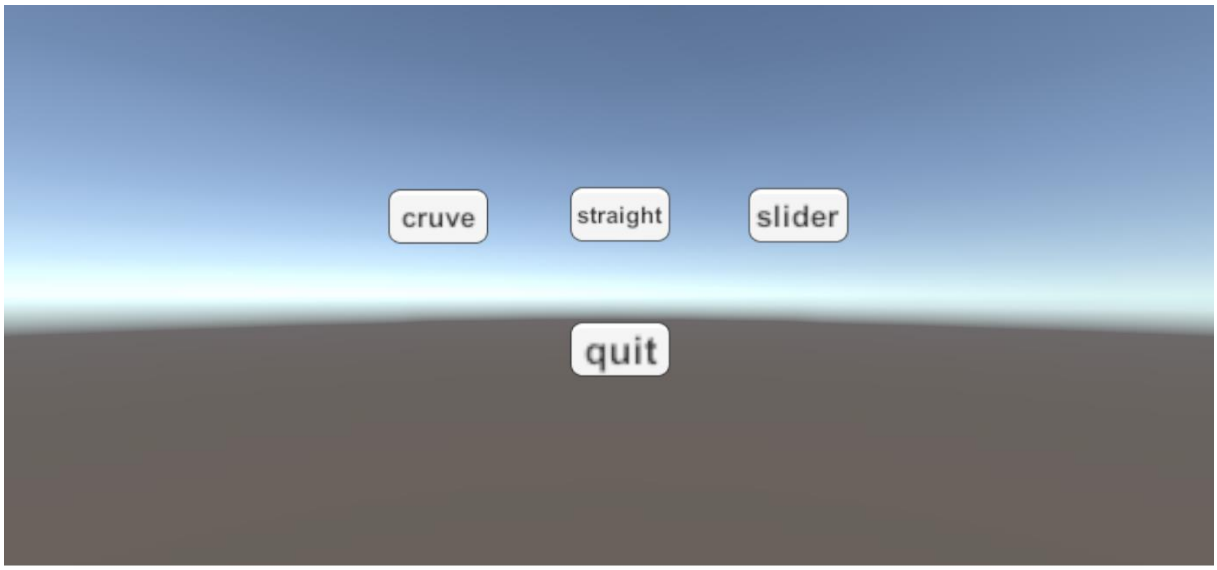


圖 43 遊戲中使用者介面

接下來就進入了遊戲畫面，使用者可以開始進行打擊，並在打擊完成後按下搖桿的選單介面按鈕回到使用者介面，後按下 quit 按鈕以中止遊戲。

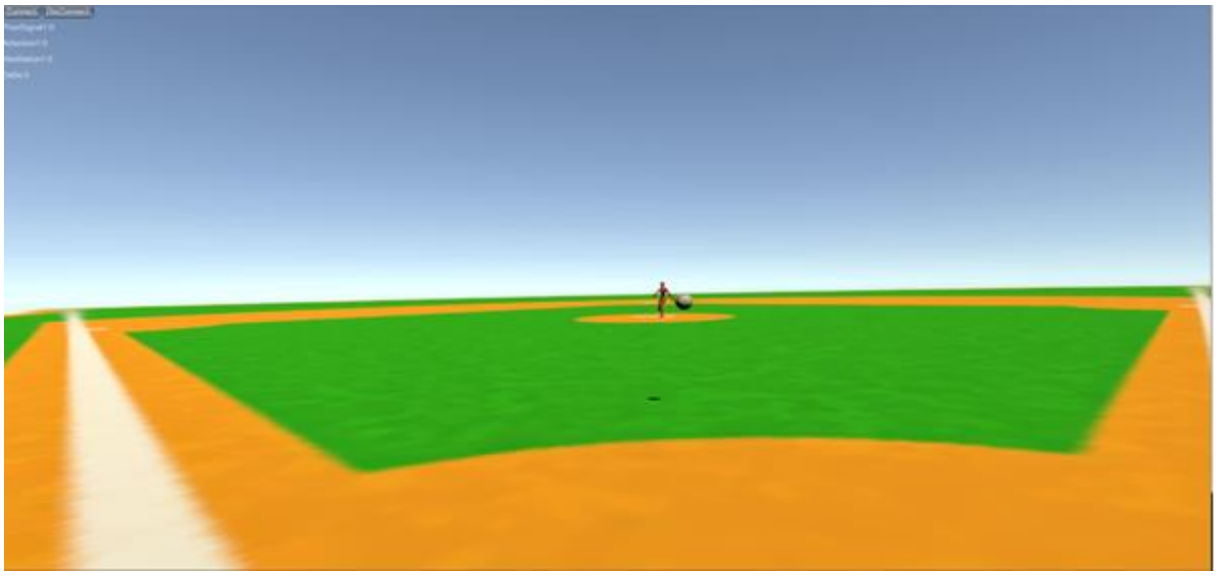


圖 44 遊戲畫面

第六章　結論與未來展望

結論:

經過這次專題讓我們了解到 VR 的製作不是我們想像中的那樣簡單，首先設備的有無、齊全，就是一個問題，幸虧我們的實驗室裡就有齊全的設備能讓我們安心使用，不過在剛開始接觸時，因為設備設定耗費了一些時間多虧了學長姊的教學，讓我們知道了設備的知識，例如：從最基本的手把的按鈕名稱到無線基地台的運作原理，而製作 VR 是必須要用到 Unity，完全不懂 Unity 是甚麼的我們為此去查找資料、借閱書籍、詢問學長姐們的意見，透過不斷的學習鑽研後，了解了如何製作場景，新增物件，設定參數，為物件增加屬性等等… 當然在製作過程中也遇到許多問題，例如：當時球與球棒無法碰撞，因為球棒是一個貼圖並無物理系統，所以無法進行碰撞，但當加上 Rigdbody(剛體)後，卻因為有了重量後會脫離手把，所以我們就有了在球棒上加上膠囊體的想法，因此改變了我們原本的方法。這些對於一開始不懂的我們來說是一個很大的進步，再加上對於程式的研究，在開始製作前我們也不了解 C#語言，也是透過不斷的查找資料，觀看教學影片，借閱書籍等等才加深了我們對 C#的了解，雖然我們對於 Unity、C#與 HTC VIVE 及腦波設備不是完全了解，還有很多不足之處，但我們經過這次專題能了解該如何使用並能正常運行，也算是達成了心目中的期待。

未來展望

未來希望能夠加入遊戲建模場景，使遊戲畫面更為精緻，加入更多球種以及球棒打擊震動感以利打擊體驗，紀錄腦波數值並加以利數據分析，並給出打擊後評語，提升整體使用體驗。

參考文獻

(一)書籍

[1] 吳亞峰，索依娜，你也能完成VR場景 用Unity實作3D及虛擬實境遊戲，出版社：佳魁資訊

[2] 賴祐吉, 姚智原，人氣遊戲這樣做!Unity 3D遊戲設計實例講堂，出版社：旗標

[3] 彭建文，C#程式設計從入門到專業(上)：完全剖析C#技術實務，出版社：博碩

[4] 彭建文，C#程式設計從入門到專業(下)：職場C#進階應用技術，出版社：博碩

網站

[1]四種實境 - VR、AR、SR、MR

https://benevo.pixnet.net/blog/post/63012046-%E5%9B%9B%E7%A8%AE%E5%AF%A6%E5%A2%83---vr%E3%80%81ar%E3%80%81sr%E3%80%81mr

[2]虛擬實境裝置HTC VIVE 硬體介紹

https://forum.gamer.com.tw/C.php?bsn=60599&snA=8760

[3]UNITY Scripting API

https://docs.unity3d.com/ScriptReference/Accessibility.VisionUtility.html

[4]Unity3D Virtual Reality Physics – Baseball 教學影片

https://www.youtube.com/watch?v=j19QDOn2cCc

[5]Unity3D Virtual Reality Physics – Baseball 教學影片

https://www.youtube.com/watch?v=FNi-UyIJV4A&t=330s

[6]神奇的意念控制設備：BrainLink Lite意念力頭帶

https://kknews.cc/tech/rz9r94x.html